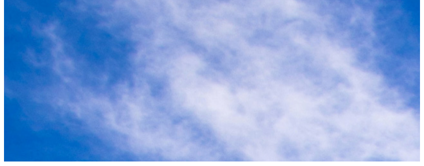
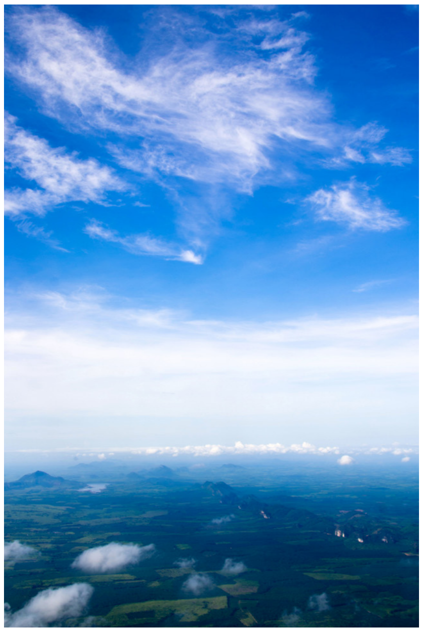
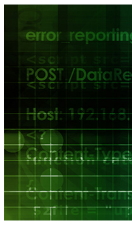


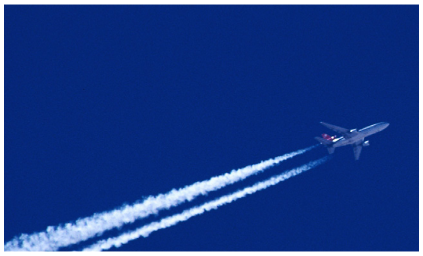
April 2016

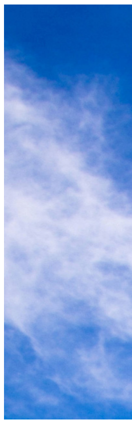
MITRE

FEDERAL AVIATION ADMINISTRATION AGILE ACQUISITION PRINCIPLES AND PRACTICES









Avinash Pinto • Michael E. Liggan • Nadya K. Subowo • Hugh G. Goodwin • Siroos Sekhavat-Tafti • Amanda M. Staley

EXECUTIVE SUMMARY

The Federal Aviation Administration (FAA) operates in a challenging environment to provide the safest and most cost-effective air traffic management system in the world. To address the challenge, the FAA acquires and operates a wide variety of information technology systems, from internal administrative systems, similar to commercial businesses, to unique safety-critical air traffic control systems. Too often, the development and implementation of these systems has involved cost and/or schedule overruns that detract from the anticipated operational value, or involve times to acquire or develop that are not sufficiently responsive to the needs of a changing environment.

As part of its commitment to continuous improvement to enhance acquisition outcomes, the FAA is pursuing the use of Agile acquisition practices to accelerate the deployment of capabilities and to ensure that capabilities exhibit greater user acceptance and operational value. Since Agile emerged as a software development methodology in 2001, it has matured to become the leading software development methodology in commercial industry. Agile has been used in critical mission software for banking, healthcare, and automotive industries. Agile has growing adoption in agencies across the federal government to deliver capabilities rapidly and iteratively to users and to realize the value of acquired capabilities. The FAA is currently exploring how to best integrate Agile practices into the Acquisition Management System (AMS).

This document represents an initial step in implementing Agile practices in FAA acquisitions, where appropriate. The identified principles and practices provide insight to FAA acquisition executives regarding implementing Agile in the FAA environment, identifying differences between Agile and traditional approaches and the prospective value of Agile. Among the considerations addressed are Agile influences on planning, requirements management, contracting, cost and effort estimation, test and evaluation, deployment, enterprise architecture, and program management. The principles and practices include the vital topic of developing an Agile culture and organization. An approach to applying Agile acquisition to AMS is suggested.

The FAA is currently in the early phases of adopting Agile, and this document provides the key principles and practices that should be considered as it integrates this new approach. Next steps in Agile adoption include:

- Assess organizational readiness and the prospective challenges involved in pursuing Agile.
- Train the workforce to understand and apply Agile principles and practices, including modifying practices as appropriate in response to the needs of the environment.
- Pursue an Agile Pilot program to validate practices adapted to the FAA.
- Evaluate and adopt emergent system engineering practices that enable and complement Agile development.

ACKNOWLEDGMENTS

This document is the product of a collaboration between the Federal Aviation Administration (FAA) and MITRE's Mission-Oriented Investigation and Experimentation (MOIE) program, an independent research and development (R&D) program sponsored by the FAA. The MOIE program is focused on R&D for sponsor mission applications, and this project was selected due to its relevance and potential impact on FAA acquisition and systems engineering.

This document has been adopted by the FAA, and is available on the FAA Acquisition System Toolset (FAST) website at: http://fast.faa.gov.

The authors are grateful to several individuals from MITRE for their subject matter expertise, review, and support. These include:

- Craig Wanke
- Andy Anderegg
- Urmila Hiremath
- Dennis Sawyer
- Dean Lamiano
- Greg Howard
- Pete Modigliani
- Su Chang
- Awais Sheikh
- Sadia Syeda
- Duncan Thomson
- Frank Willingham

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

1

1.1 Purpose

The Federal Aviation Administration (FAA) operates in a challenging environment to provide the safest and most cost-effective air traffic management (ATM) system in the world. To address the challenge, the FAA acquires and operates a wide variety of information technology (IT) systems, from internal administrative systems, similar to commercial businesses, to unique safety-critical air traffic control systems. Too often the development and implementation of these systems has involved cost and/or schedule overruns that detract from the anticipated operational value, or involve times to acquire or develop that are not sufficiently responsive to the needs of a changing environment.

As part of its commitment to continuous improvement to enhance acquisition outcomes, the FAA is pursuing the use of Agile acquisition practices to accelerate the deployment of capabilities and to ensure that capabilities exhibit greater user acceptance and operational value. Since Agile emerged as a software development methodology in 2001, it has matured to become the leading software development methodology in commercial industry. Agile has been used in critical mission software for banking, healthcare, and automotive industries. Adoption of Agile is growing in agencies across the federal government to deliver capabilities rapidly and iteratively to users and realize the value of acquired capabilities. The FAA is currently exploring how to best integrate Agile practices into the Acquisition Management System (AMS).

This document represents an initial step in implementing Agile practices in FAA acquisitions, where appropriate. The identified principles and practices provide insight to FAA acquisition executives regarding implementing Agile in the FAA environment, identifying differences between Agile and traditional approaches and the prospective value of Agile. Among the considerations addressed are Agile influences on planning, requirements management, contracting, cost and effort estimation, test and evaluation (T&E), deployment, enterprise architecture, and program management. The principles and practices address the vital topic of developing an Agile culture and organization. An approach to applying Agile acquisition in AMS is suggested.

1.2 Document Organization

This document describes the principles, practices, and challenges associated with an FAA Agile Acquisition process. Section 1.1 has discussed the purpose of this document. The remainder of this document is organized as follows:

The remainder of Section 1 provides background on the Agile process (Section 1.3), and the status of Agile adoption in government (Section 1.4).

To effectively implement Agile within the FAA, the canonical AMS lifecycle needs to be adapted. Section 2, Tailoring Acquisition Management System for Agile Acquisition, provides guidance for determining when

Agile best applies to a program and suggests a notional Agile lifecycle model.

Section 3 addresses the key aspects involved in applying Agile within the AMS. These aspects include the following:

- Agile Culture and Organization (Section 3.1) discusses the roles and responsibilities for executing an Agile Acquisition.
- Agile Planning (Section 3.2) presents the evolution of current strategic and tactical planning activities toward one that facilitates continuous improvement.
- Agile Requirements (Section 3.3) details a requirements solicitation and management framework that promotes delivery of valuable products in an iterative manner.
- Agile Cost and Effort Estimation (Section 3.4) identifies the changes to current cost and effort estimation processes when accommodating an Agile development process.
- Agile Contracting (Section 3.5) presents the most effective contracting strategies suitable for executing an Agile Acquisition.
- Agile Test and Evaluation (Section 3.6) discusses a re-conceptualized Agile T&E framework.
- Agile Deployment and Sustainment (Section 3.7) identifies an Agile approach that promotes continuous development and management based on technical excellence and responsiveness to users.
- Agile Program Management (Section 3.8) presents an approach for the necessary organization and administration activities when executing an Agile Acquisition.
- Enterprise Architecture-Transition to Agile (Section 3.9) highlights the Agile principles that may be adopted when developing the enterprise architecture, including any supporting artifacts.
- Next Steps (Section 4) concludes with the identification of follow-on activities in preparation for developing a more comprehensive execution plan.

1.3 Agile Fundamentals

1.3.1 Agile Origins

The Agile methodology is a philosophy and collection of techniques that promote adaptive planning, evolutionary development, early and iterative delivery of functionality and operational value, and continuous improvement through user feedback loops. The Agile process is conducted through the collaboration of self-organizing, cross-functional teams and provides an alternative to traditional project management and development approaches (e.g., Waterfall, Spiral, or the V-model).

In response to the increasing potential for software development project failures, as documented in studies such as the Standish Group CHAOS reports [1], software development researchers/practitioners defined alternatives adapted from incremental approaches and Lean principles that have come to be known as "Agile." The term was coined in February 2001 by a group of software developers meeting at the Snowbird Resort in Utah to discuss lightweight software development methods. These methods were being evolved in response to increasingly cumbersome and document-driven development processes perceived to be impeding software development. The fundamentals of Agile were captured in the publication of the Manifesto for Agile Software Development [2], which expresses the values and principles of the Agile development approach. Agile development has become increasingly popular in the software development community and continues to expand in scope and application to larger developments, to more challenging environments such as government IT procurement, and to the conduct of project activities not directly related to software.

The Agile manifesto expresses four values and 12 principles, depicted in Table 1 1. The bolded items under Agile Development Values represent items of the highest value.

Table 1-1 | Agile Values and Principles

AGILE DEVELOPMENT VALUES	AGILE DEVELOPMENT PRINCIPLES
1. **Individuals and interactions** over Processes and tools 2. **Working software** over Comprehensive documentation 3. **Customer collaboration** over Contract negotiation 4. **Responding to change** over Following a plan	1. Customer satisfaction by rapid delivery of useful software 2. Welcome changing requirements, even late in development 3. Working software is delivered frequently (weeks rather than months) 4. Close, daily cooperation between business people and developers 5. Projects are built around motivated individuals, who should be trusted 6. Face-to-face conversation is the best form of communication (co-location) 7. Working software is the principal measure of progress 8. Sustainable development, able to maintain a constant pace 9. Continuous attention to technical excellence and good design 10. Simplicity—the art of maximizing the amount of work not done—is essential 11. Self-organizing teams 12. Regular adaptation to changing circumstance

1.3.2 Differences between Agile and Waterfall

Figure 1 1 summarizes the primary differences between the Agile approach and the commonly employed Waterfall approach.

The Waterfall model is a sequential process in which progress flows through sequential phases of conception, initiation, analysis, design, development, testing, production/implementation, and sustainment. Conceptually, each phase is completed before moving to the next phase. The model is considered to be derived from the manufacturing and construction industries and was applied to software development in the absence of software-specific alternatives. The process shuns change in subsequent phases, emphasizing the need for comprehensive planning up front.

In contrast, the Agile model is iterative, with progress advancing incrementally in small units of development activity, which deliver complete functions. Each iteration of development activity involves aspects of analysis, design, development, testing, and demonstration. User feedback is provided in each iteration to better ensure the operational suitability of the evolving product (e.g., a capability or service). Rather than focusing on comprehensive planning and shunning change, the Agile model involves expecting and adapting to change. It involves doing "just enough" planning to execute the intended development through just-in-time decomposition and decision making. "Just enough" details are developed to maintain progress (avoids over-engineering), and commitment-level decisions are deferred until ready to execute.

AGILE	SUBJECT	WATERFALL
Elaborated throughout – Expects that requirements cannot be perfect from the start	**Requirements**	Fully defined and well understood Up-Front
Highly collaborative	**Management Structure**	Command and control
Short, iterative cycles	**Release Schedule**	Single release
Expected part of every iteration	**Perspective on Change**	A deviation from the baseline requiring a formal change request
Focus on flexibility and value delivery	**Delivery**	Focus on requirements
Indefinite delivery, indefinite quantity, time-and-materials, short term task orders	**Contracting Vehicle Orientation**	Firm fixed price contracts
Important, but developed as needed – just-in-time	**Documentation**	Documentation is critical and required up-front

Figure 1-1 **Agile and Waterfall Differences**

Unlike the Waterfall model, in which requirements are fixed and degrees of freedom are often the cost and schedule, the Agile model allows requirement changes and scope variability based on user feedback or if issues impede function development. The Agile model adheres to schedule, applies resources consistently (i.e., an established development capacity), and focuses on value delivery, as represented in Figure 1 2. Prioritization of the requirements promotes realizing the most valuable capabilities first and deferring or eliminating developments of lesser value, if necessary.

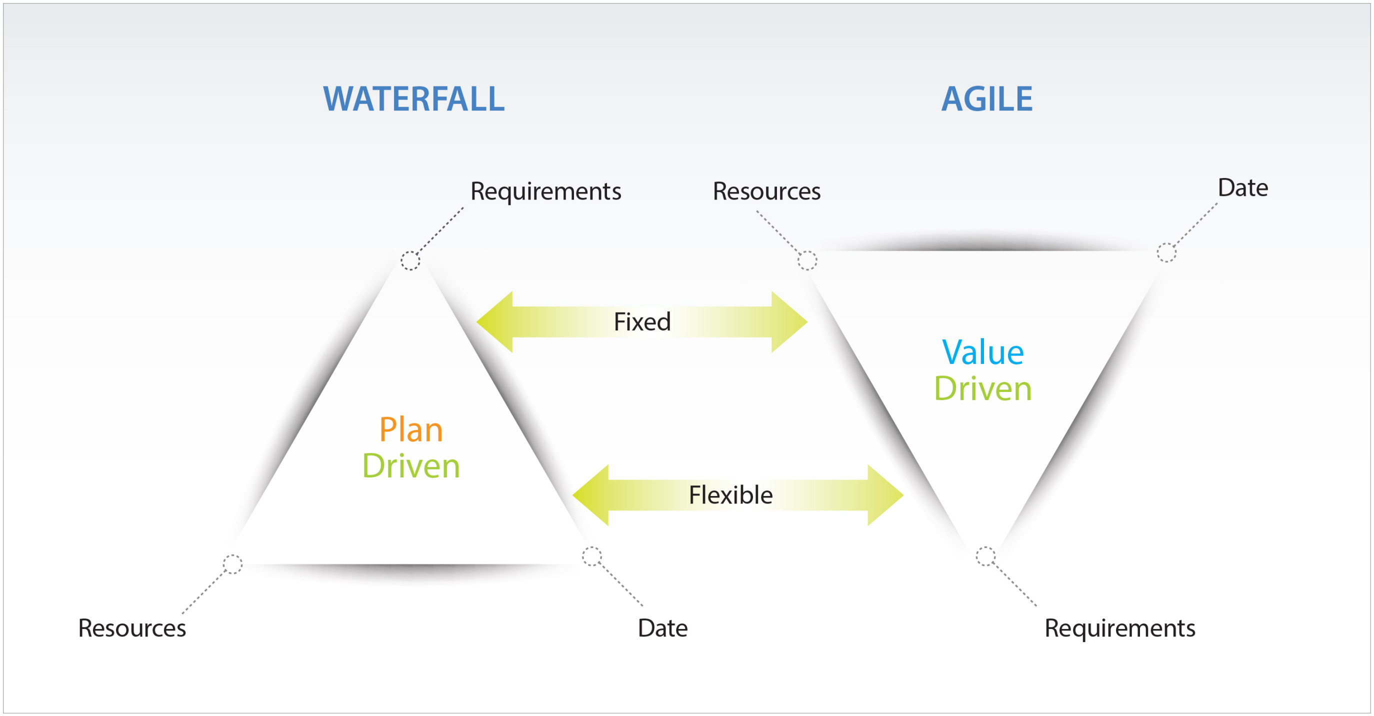


Figure 1-2 Fixed vs. Flexible in Waterfall and Agile

1.3.3 Agile Process and Framework

The Agile model and philosophy have been instantiated in a number of branded methods with varying focus and techniques. Scrum is the most popular and widely applied. In the sections that follow, the Scrum method has been adapted for use in characterizing an Agile framework suitable for application to government programs. The terminology utilized in this Agile framework is consistent with the terms used by government programs applying Agile Scrum methods.

The Agile framework consists of events, roles, and artifacts. Figure 1 3 presents some of the core elements of an Agile framework. This section details the events, roles, and artifacts.

Agile events include the following:

- **The Sprint:** In Agile, software/system capabilities are developed through a series of Sprints, a time-boxed period during which a potentially releasable "working capability" is developed. The lengths of Sprints (e.g., days to several weeks) are constant for a particular program. The Sprint duration selected depends on the implementing organization and the overall complexity of the development, test, and production environments. This defined timeframe assigned for each planned development activity is called "time-box." Time-boxing is limiting the amount of time conducted on an activity, and is a critical aspect of the Sprint.
- **Planning:** The planning occurs at the Program, Release, and Sprint levels. Program planning is more strategic in nature and includes defining the product vision and roadmap consistent with enterprise objectives, and capturing and prioritizing the operational needs in the Program Backlog. Release planning also is more strategic and focuses on how the program can incrementally deliver value

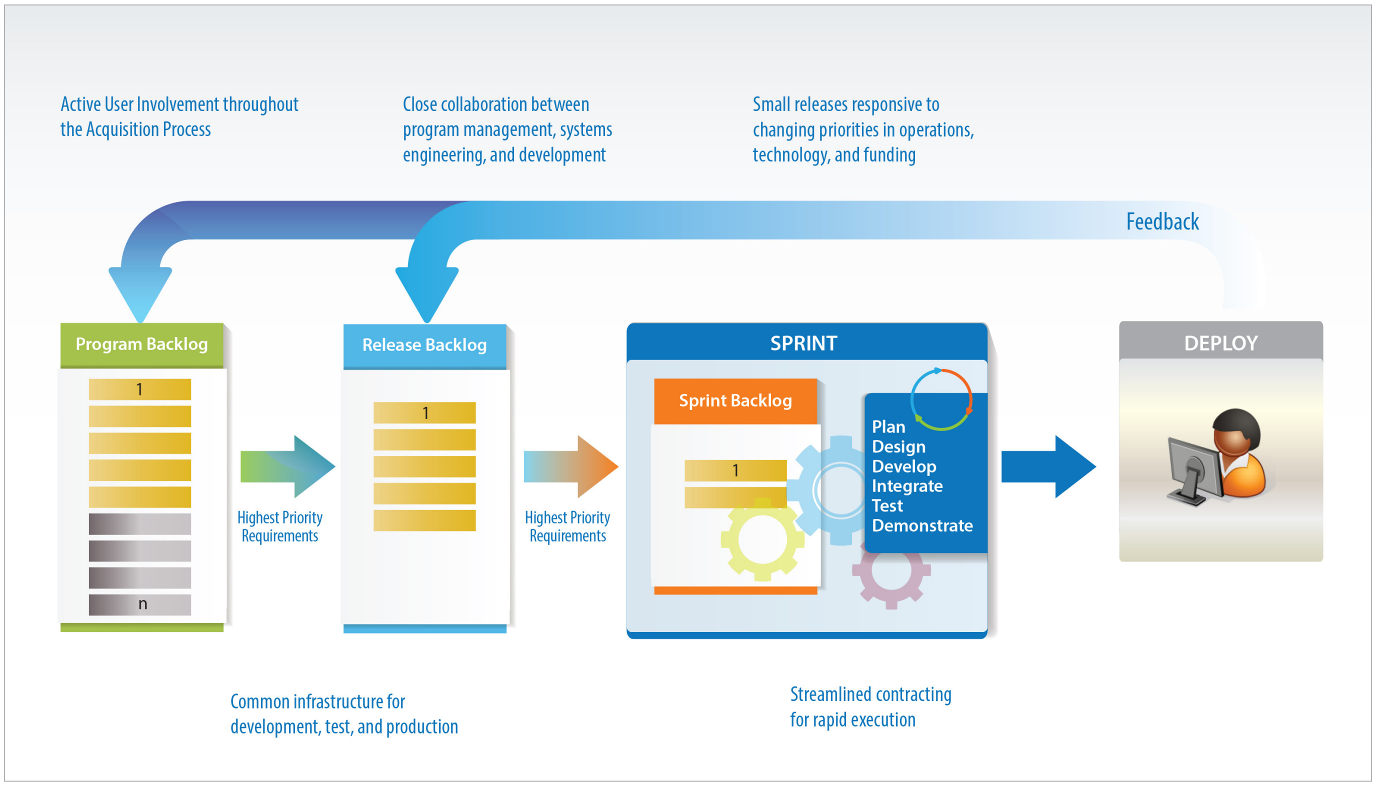

Active User Involvement throughout the Acquisition Process
Close collaboration between program management, systems engineering, and development
Small releases responsive to changing priorities in operations, technology, and funding
Feedback
Program Backlog
1
n
Highest Priority Requirements
Release Backlog
1
Highest Priority Requirements
SPRINT
Sprint Backlog
1
Plan
Design
Develop
Integrate
Test
Demonstrate
DEPLOY
Common infrastructure for development, test, and production
Streamlined contracting for rapid execution

Figure 1-3 Agile Development Process

through major capability releases. During release planning, prioritized features from the Program Backlog are selected for a particular release. Release planning occurs at a cadence commensurate with the duration of each release development. At a Sprint level, planning is typically time-boxed to a day for a one-month Sprint. Longer or shorter Sprint durations will have longer or shorter planning times, respectively. Sprint planning entails two primary objectives: 1) identifying the list of features to be developed during a particular Sprint (as selected from the Release Backlog), and 2) understanding the work required to develop the features.

- **Daily Scrum:** The Daily Scrum is the heart of the Agile process, and is typically a 15-minute event during which the Development Team coordinates and plans their activities for the next day. It is not intended to be an event to foster a lot of discussion, but rather one in which progress is shared and measured, and obstacles are identified. Three fundamental questions are asked of each Development Team member during the Scrum:
  - What did you do yesterday to contribute toward the Sprint?
  - What will you do today to contribute toward the Sprint?
  - Are there any impediments that are keeping you from making progress?

  Following the Daily Scrum, team members meet to discuss issues at greater length, or to re-plan their activities as necessary.
- **Review:** Reviews occur at the Sprint and Release levels. The purpose of the Sprint Review is to provide a forum that allows the Agile team (Product Owner, Development Team, Scrum Master), the

acquisition team, and stakeholders to understand what was accomplished during the Sprint, relative to the Sprint plan and Release Backlogs. During the review, the developed capability is demonstrated to program stakeholders. Feedback received during the Sprint Review is used for subsequent Sprint Planning. The Release Review serves a similar function, but it is a more prominent event as the capabilities encapsulated in the Release will be delivered to the user community. The Release Review has broader involvement, and includes stakeholders from interacting programs in addition to the user community.

- **Retrospective:** Retrospectives occur at the completion of any significant Agile event—at the end of a Sprint, Release, or Program. Retrospectives are an outcome of a "continuous improvement" mind-set. The purpose of the Retrospective is to evaluate the event in terms of people relationships, processes, and tools. It also includes a discussion of what went well, issues that arose, and resolutions that were made. Based on the evaluation, improvements are identified and prioritized for implementation in successive events.

Agile employs the following specific roles as part of its framework:

- **Product Owner:** The Product Owner is responsible for maximizing the value of the work conducted by the Development Team. The Product Owner is responsible for the backlog, and prioritizing the capabilities in the backlog in order of optimal business value. The Product Owner also works with the Development Team to ensure that the product backlog is well understood. Note that the term "product" is used loosely to represent a variety of solution types. It is meant to describe a given solution to be acquired to address a mission need and does not always imply automation. Products can be systems, services, procedures, policies, capabilities, or functions.
- **Development Team:** The Development Team is responsible for conducting the development work necessary to deliver a potentially releasable increment of functionality at the end of each Sprint. Development Teams are a holistic construct, and do not contain sub-teams. Development Teams tend to have a combination of generalized and specialized expertise, which allows teams to maintain stability across Sprints. Teams may be augmented with specialty engineers, who focus on security, network management, data management, etc. They are cross-functional in nature, empowered to organize themselves and decide how best to complete the work necessary for each Sprint. Ideally, these teams contain no more than 7 +/- 2 members, in order to accomplish a significant amount of work while keeping coordination overhead to a minimum.
- **Scrum Master:** The Scrum Master is responsible for maintaining the structure and the Scrum rules of operation. The Scrum Master provides a "servant-leader" function to the Product Owner and the Development Team. The Scrum Master works with the Product Owner on managing and prioritizing the backlog, and works with the Development Team to maximize the quality of work accomplished and ensure that obstacles to progress are removed. The Scrum Master serves a coaching function for the Development Team's members, and ensures that Agile concepts and practices are well understood and applied.

Agile leverages the use of key artifacts as part of its framework, including a backlog and a set of roadmaps. From a government perspective, these backlogs and roadmaps must be framed and understood within the context of a program. Key artifacts include the following:

- **Program Backlog:** Comprises the set of functional and non-functional requirements to be delivered by the Program. It contains a prioritized list of features, which are the key capabilities or services that will be provided by the system that is being developed. The Program Backlog is never complete—it initially identifies the best understood requirements, and subsequently evolves as the use for the system and the understanding of the operating environment evolve.
- **Release Backlog:** Comprises the subset of Program Backlog requirements that has been selected for the development of a specific Release. It contains a prioritized list of features, or services that will be provided by the system during the next delivery to the user.
- **Sprint Backlog:** Comprises the subset of Release Backlog items that has been selected for development for a given Sprint. It is owned and maintained by the Development Team and encapsulates all the work necessary to complete the Sprint. The Sprint Backlog reflects a real-time view of the completed and planned work in a Sprint, and is revised accordingly during a Sprint to maintain accuracy.
- **Roadmap:** Describes the planned evolution of a product. It consists of smaller development efforts (Sprints), and aligns the development schedule to business objectives.

An important element of consideration within the Agile process is the notion of requirements. Requirements within Agile are written at different levels, depending on the following perspective:

- **Epics:** Epics are a large user story, often defined for a release that spans multiple Sprints. Epics may reflect enterprise (business and architectural) level initiatives that are substantial in scope. Business epics are user-facing initiatives that are intended to realize new opportunities and deliver benefits to the user community. Architectural epics are technology initiatives necessary to evolve portfolio solutions to support current and future business needs.
- **Features:** Features are program-level requirements that represent the key capabilities or services provided by a system to fulfill stakeholder needs. Features are typically maintained in the Program Backlog.
- **User stories:** User stories are Sprint-level requirements that describe the functionality desired by a user and the non-functional requirements that act as constraints. User stories are written in a structured format and maintained in the Sprint Backlog. A key aspect to consider for the User Story is the "definition of done," when the development of a capability is complete. Several factors may influence when a capability is "done," including: unit, integration, and acceptance tests have been passed; acceptance criteria have been met; and the capability has been accepted by the Product Owner.

In addition to the events, roles, and artifacts, the Agile framework also introduces new terminology regarding the development process. Terms include the following:

- **Cadence:** Refers to development at a sustainable pace, where there is a consistent flow of activities (e.g., planning and deployment and retrospective) within the time-box.
- **Velocity:** Represents the rate at which functionality is being delivered. It may be measured in terms of story points per Sprint, where story points are a relative unit of measurement to describe the complexity of user stories.

- **Burn-down chart:** Graphically captures the amount of work left to do versus time. The chart highlights the time left until completion of all work. It is a helpful tool for estimating when the work will be accomplished, and can be used to estimate whether progress is ahead of or behind schedule.

1.3.4 Scaling Agile

In recent years, the success of Agile has resulted in the development of frameworks aimed at applying Agile techniques to efforts of increasingly larger scope, projects of significant complexity involving more than just Development Teams, and efforts outside the IT domain. A few of the more mature frameworks include the following:

- **Disciplined Agile Delivery (DAD):** DAD is a process decision framework that encompasses strategies and best practices from several other models. DAD emphasizes people first, is goal driven and learning oriented in nature, and accommodates several system delivery lifecycles, including Agile software delivery, Lean/Kanban, and a continuous delivery lifecycle. [3]
- **Large-Scale Scrum (LeSS):** LeSS applies Scrum to several teams working together on a single product through two frameworks—LeSS and LeSS Huge. The latter scales to more than eight teams and can accommodate up to a few thousand people. This framework applies the principles, elements, and purpose of Scrum within a large-scale context. With LeSS, an organization-specific framework is developed and applied, based on the product, processes, and organization design. [4]
- **Scaled Agile Framework (SAFe):** SAFe emphasizes an enterprise-wide approach to Agile, which aligns activities at the Team, Program, and Portfolio levels. The framework addresses governance, organizational, and technical processes, and their alignment across each level. SAFe promotes the identification of business opportunities and architecture evolution at the Portfolio level, which are then managed at the Program level and developed at the Agile Team level on a regular cadence. [5]
- **Scrum-of-Scrums (SoS):** SoS provides the means for an organization to apply Scrum at scale. In SoS, multiple teams work together on the same product, with each having daily Scrum sessions to coordinate activities. The timing of the Scrum meetings is aligned in order to offer the Scrum Master (or another member) of each team to have a "Scrum of Scrums" meeting, which serves to coordinate activities across multiple teams. SoS is an approach that has been adopted by other frameworks aimed at scaling Agile techniques. [6]

1.3.5 Agile Acquisition

Whereas the roots of Agile began in software development, Agile techniques have broadened to help programs acquire capabilities more expeditiously. Agile Acquisition is a strategy for providing multiple, rapid deliveries of incremental capabilities for operational use and evaluation. It embraces the values and principles outlined in Agile and adapts them to the acquisition process to facilitate rapid development and delivery of business value. Agile Acquisition approaches comprise several facets of the acquisition process, including requirements, systems engineering, contracting, costing, development, integration/testing, deployment, and sustainment. The strategy assumes a predominately software-based development of the system.

In an Agile Acquisition process, elements of the organization are aligned in order to support Agile techniques. At the core, these elements comprise the Program Manager, system engineers, user community representative(s), and the development and test team. Other key elements needed to support the Agile process include the acquisition office, contracting officer, operational leadership, costing and financial support, and enterprise architects. The user community is an important element that needs to be defined early in the process and requires ongoing engagement. Within the FAA, the user community for a program may include FAA Lines of Business, FAA personnel, the airlines, and general aviation.

Once the organizational elements are identified and defined with respect to their role in the Agile Acquisition process, it is important to tailor the program structure to facilitate Agile Acquisition. Ideally, the program is structured to realize the delivery of capabilities on a regular basis, without emphasizing the need for extensive up-front documentation. Acquisition practices and functions are assessed and evaluated to accommodate the appropriate Agile techniques required to facilitate expeditious delivery.

Some of the overall principles behind Agile Acquisition include the following:

- **Structure the program to deliver usable capabilities every six to 12 months:** Time-boxing the delivery of capabilities (for development and deployment) every six to 12 months helps to mitigate the risk of technology irrelevance or obsolescence prior to deployment. It also serves to frequently align the delivered capabilities with emergent priorities and reduce integration risks. A key aspect of this approach is to require the developer to deliver regular iterations of capability for evaluation and/or deployment, as appropriate. This allows the government program, stakeholders, and user community to provide frequent feedback into the development process, to ensure that the capabilities meet user needs. It also allows for timely verification and validation, and test and evaluation.
- **Prioritize and evolve requirements in alignment with emergent needs:** Instead of a formal series of static requirements documents, requirements in Agile are typically managed through a backlog of needs, where the backlog represents an evolving requirements baseline. The program actively works with the user community and other stakeholders to identify the highest priority needs, which are developed during the next available Sprint. The system engineers and architects work with the program to ensure that the necessary infrastructure functionality is developed to support existing and future capabilities. The backlog of needs is prioritized based on value and duration—those that deliver the highest value and have the shortest duration are developed first. After a Sprint, if a capability has been deemed good enough to meet user needs, development shifts to the next priority.
- **Leverage common infrastructure for development, test, and production:** Agile techniques emphasize the use of common infrastructure platforms to improve interoperability and security, while decreasing costs and time required for deployment. Leveraging enterprise services (e.g., security, data dissemination) whenever feasible reduces development time so that programs can focus on developing the core functionality required for capabilities. Automated tools enable continuous integration and testing to reduce the delays caused by errors found downstream. The common infrastructure platforms continually evolve to accommodate new capabilities and infrastructure functionality.

- **Integrate users throughout the acquisition and development process:** A key tenet of Agile Acquisition is to integrate users throughout the acquisition and development process. It is important for programs to strive to continually improve the mutual understanding between the acquisition and user communities (both internal and external to the FAA). User conferences held with all stakeholders can enhance collaboration, and facilitate clarification and consensus on the objectives and priorities of the program. Regular system demonstrations to users can be a valuable means for users to provide the necessary feedback. User priorities influence the capabilities included in successive Releases and Sprints.

1.4 Agile Adoption in the Government

Agile methodology has been widely adopted in software development. Currently, Agile is being expanded to encompass enterprise-wide concerns through methodologies such as the Scaled Agile Framework [5] and into systems engineering through the introduction of a chapter on Agile in the International Council on Systems Engineering (INCOSE) handbook [8].

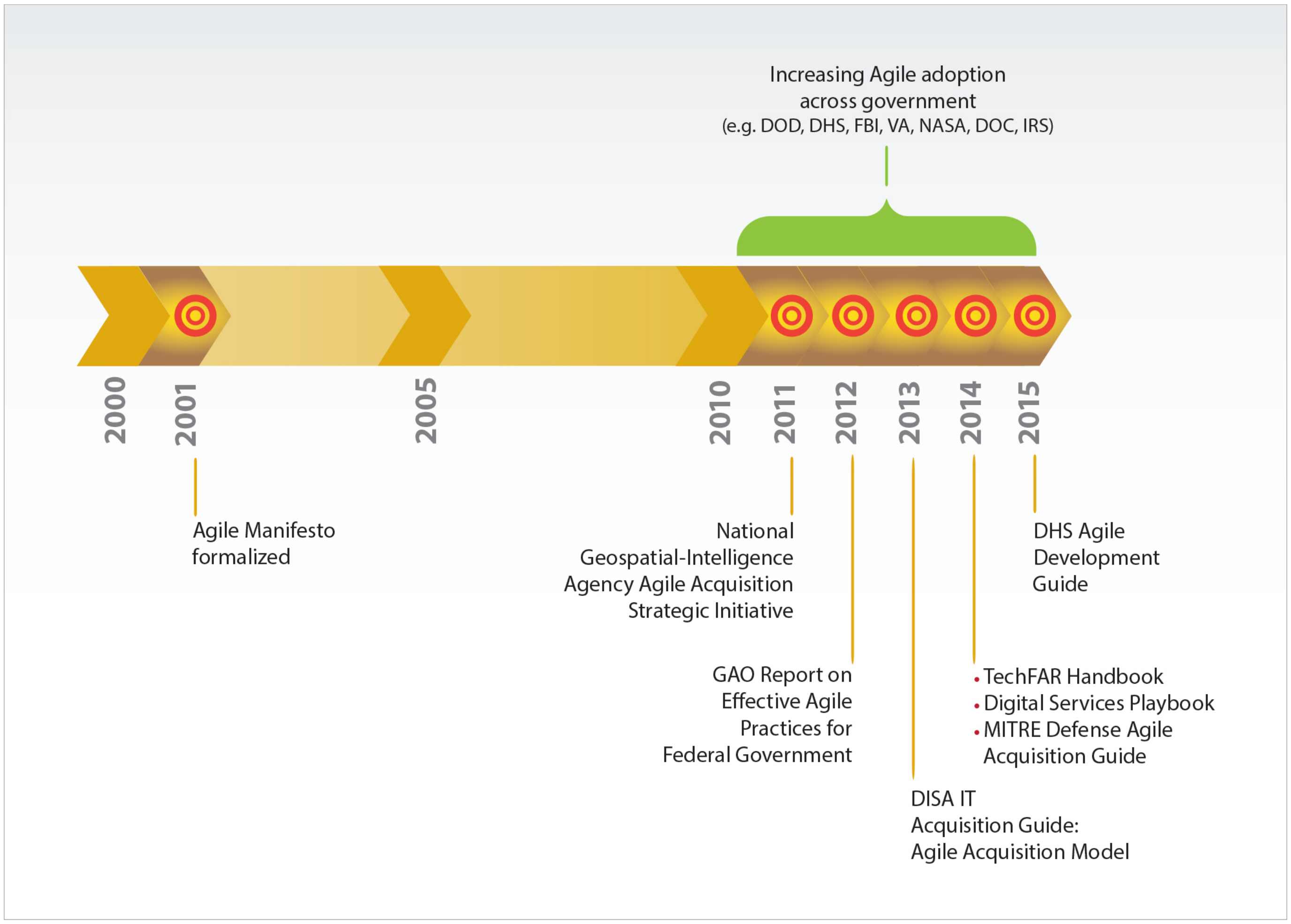

Increasing Agile adoption across government (e.g. DOD, DHS, FBI, VA, NASA, DOC, IRS)
2000
2001
2005
2010
2011
2012
2013
2014
2015
Agile Manifesto formalized
National Geospatial-Intelligence Agency Agile Acquisition Strategic Initiative
GAO Report on Effective Agile Practices for Federal Government
DISA IT Acquisition Guide: Agile Acquisition Model
• TechFAR Handbook
• Digital Services Playbook
• MITRE Defense Agile Acquisition Guide
DHS Agile Development Guide

Figure 1-4 Agile Acquisition in the Government

The Department of Defense (DoD) is implementing guidance for conducting defense acquisitions using Agile practices [7]. The White House has published a Digital Services Playbook that explicitly advises, "build the service using Agile and iterative practices" ("play" #4) [9]. The TechFAR handbook [10] highlights the flexibilities in the Federal Acquisition Regulation (FAR) that can help agencies implement "plays" from the Digital Services Playbook that would be accomplished with acquisition support—with a particular focus on how to use developers to support an iterative, customer-driven software development process, as is routinely done in the private sector.

Agile Software Development and Acquisition practices have been implemented by several United States (U.S.) federal government agencies, as shown in Figure 1 4. Successful Agile adopters include the DoD, Department of Homeland Security, Department of Veterans Affairs (VA), National Aeronautics and Space Administration (NASA), and National Geospatial-Intelligence Agency, among others. The following examples highlight the impact of Agile in the federal government:

- DoD's Global Combat Support System-Joint (GCSS-J) program began leveraging Agile practices in 2008 to better address user needs, and since then has successfully delivered capabilities to the field every six months. [11]
- By introducing Agile practices in 2010, DoD's Integrated Strategic Planning & Analysis program was able to shorten the acquisition cycle duration between program initiation and Initial Operational Capability by 45 months.
- Through utilizing Agile practices, NASA's Mission Control Technologies project was able to shorten its release cycle, with better results and lower costs than traditional development methods. Releases are delivered to the customer every three weeks, and a formal release for operations is available every three months.
- Prior to adopting Agile practices in 2009, the VA delivered IT capabilities every three to seven years. Since adopting Agile practices, it delivers capabilities every 4.2 months on average.

Agile practices have also been adopted by several international government organizations. For example, NATO successfully applied Agile to the development of a capability fielded at 20 NATO sites, with strong user involvement. The time between contract award and fielding the capability was 1.5 years.

TAILORING THE ACQUISITION MANAGEMENT SYSTEM FOR AGILE ACQUISITION

2

2.1 Introduction

The FAA AMS provides policy and guidance for FAA acquisitions [34]. The AMS policy constitutes agency-wide, mandatory requirements captured in the Acquisition Management Policy document [51]. Guidance is provided in the form of guidelines, processes, instructions, templates, databases, handbooks, checklists, and other information to execute the AMS policy [52]. This information guides and supports the workforce in planning and executing acquisition management policy and all related lifecycle management and functional discipline activities and processes. Acquisition management guidance is derived from the acquisition management policy and should be followed unless there is a rational basis for adopting a different approach.

Agile might be introduced to the AMS in several ways, including using the existing AMS tailoring mechanisms to define an Agile approach, or to define an Agile-specific subset of AMS guidance. With respect to the first, AMS provides the flexibility for organizations managing investment initiatives to request tailoring of the AMS lifecycle management process by submitting a tailoring request to the Acquisition Executive Board (AEB) [53]. The tailoring process [54] involves preparing a tailoring request form [55] defining the requested tailoring with supporting rationale and providing it to the AEB for review and approval and subsequent distribution to the Joint Resources Council (JRC). In this approach, Agile use is treated as a program-specific request. The second way of introducing Agile is to update AMS guidance to explicitly address Agile acquisition. Section 2.3 suggests an alternative Agile lifecycle with associated decision points and artifacts that might be applied in either case.

2.2 Determining When an Agile Approach Applies

An Agile Acquisition approach may apply when the prospective attributes of the acquisition and the motivations for Agile application align. Acquisition program attributes are currently defined in the AMS through investment types and acquisition categories (ACATs) [56]. First, a proposed FAA investment program will be classified into an investment type (New Investment, Technology Refreshment, Facility, Variable Quantity, or Support Contract). Second, the acquisition program will be classified into an ACAT for that investment type based on the ACAT designation criteria. Programs will be assigned to the highest level ACAT (starting with ACAT 1) in which they meet one or more of the designation criteria. Designation criteria include total facilities and equipment (F&E) costs, single-year F&E costs, operations and maintenance (O&M) costs, and other factors such as complexity, risk, political sensitivity, safety, and security.

Based on current interpretation and application of Agile methodologies, Agile may not apply to all investment types. Agile may not apply to facility initiatives, which involve traditional facility design and construction activities. Agile may not apply to variable-quantity programs that involve additions to and/or modernizations of previously fielded systems nor to technology refreshment programs, which by definition do not include new or improved capabilities.

Agile is most suitably applied to new investments, particularly those that are software intensive. Agile originated in the software development community, and programs with significant software development continue to be the primary target of Agile use. Agile may also be applied effectively to software maintenance programs such as the software development and support service contracts applied to National Airspace System (NAS) automation systems after deployment. Although more typically resourced through O&M budgets, these programs involve significant amounts of software development for problem resolution and capability enhancement that are prioritized and task-directed by second level engineering organizations. Although not necessarily involving significant degrees of software development, non-material programs might adapt Agile approaches (e.g., airspace design or procedure development) for which software product analogies may be defined and the program motivations share similarities with Agile.

ACAT level is another program attribute influencing the potential applicability of Agile. Although Agile scaling frameworks have been and continue to be defined to apply Agile principles and methods to programs of larger size and to enterprise-level portfolios, the more traditional application is to smaller development efforts within the scope of one or several Development Teams. This would suggest that smaller programs, reflected in higher assigned ACAT levels, are better aligned with Agile approaches, pending maturing Agile capabilities within the organization.

Figure 2 1 illustrates the increasing potential applicability of Agile based on AMS investment type and acquisition category. Green represents the region best aligned with Agile use. Best alignment exists between new investments and support services of smaller size that involve software-intensive development.

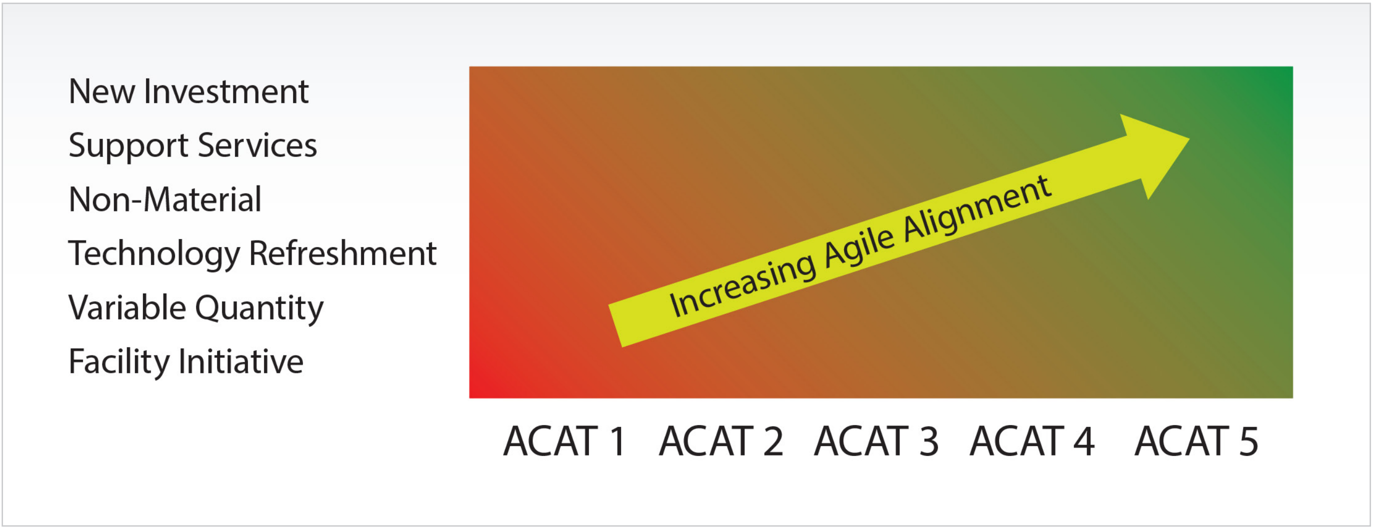


Figure 2-1 Increasing Alignment of AMS Investment Types and Categories with Agile

A decision to use an Agile Acquisition approach should also involve alignment of the program with Agile motivations. The motivations for Agile include the following:

- Business agility: There is a need or desired opportunity to streamline processes and decision making. There is a need to respond to evolving requirements or a changing environment. There is a need or desire for more opportunities to inject innovative acquisition or development approaches.
- Rapid delivery: There is a need or desired potential to deliver capabilities more rapidly without diminishing product quality.
- Sustainable delivery: There is a need or desire to provide a consistent, sustainable pace of development that makes development more predictable (i.e., no "death marches").
- Plan adherence: There is a need or desired opportunity to promote schedule and resource adherence, exercising flexibility with respect to function/feature scope as a potential trade-off. Short planning and development cycles inherent in Agile approaches promote accurate planning and measured progress.
- User acceptance: Iterative development through Sprints and Releases provides opportunity for frequent user feedback and enhanced product operational suitability, decreasing the risk in solution delivery.
- Cost-effectiveness: Providing and maintaining focus on the needs/capabilities/ features of greatest importance reduces unnecessary work and rework. Frequent feedback opportunities support continuous process improvement. Pursuing "just enough" documentation reduces the potential overhead of documentation that has little utility.
- Team satisfaction: The dedicated and self-directed Agile teams associated with Agile development promote employee satisfaction through empowerment and workload balance. Trust is implicit in the Agile process. Transparency and communication are promoted.

To summarize, a prospective acquisition program should consider an Agile approach when the program possesses attributes and motivations that align with Agile. This alignment is reflected in responses to the key questions for consideration shown in Table 2 1. Questions for Agile Consideration. Questions identified as "factor" relate to intrinsic properties of the intended acquisition; they must be accommodated. Questions identified as "plan" are those subject to influence by designing a program for Agile acquisition.

Table 2-1 | Questions for Agile Consideration

AGILE EFFORT SUITABILITY & SUCCESS PROSPECTS QUESTIONS	FACTOR/PLAN	NOTES
Culture/Organization Aspects		
Is there executive sponsorship for the effort?	Factor	In many if not most organizations, introduction of Agile involves a cultural change. To overcome the impediments to change that are likely to occur, it is important that the leadership of the organization is visibly supportive of the effort.
Does the project sponsor understand and accept the Agile philosophy?	Factor	Alignment of sponsor expectations and Agile program conduct is necessary to ensure sponsor support.
Is there a commitment to provide end-user involvement?	Factor	Customer collaboration and user feedback are fundamental to Agile methodology, involving an organizational commitment to provide the resources of user time and attention.
Is there a supportive relationship among stakeholders?	Factor	Stakeholders should understand and be supportive of the Agile approach, such as incremental addition of capabilities.
Can the organization accommodate an Agile cadence?	Factor	Agile methodologies suggest sprint durations of 2-4 weeks and program increment/release durations of 12-24 weeks, but the Agile philosophy promotes adaptability. The selected cadence must conform to the needs of the organization.
Is the Project Manager (PM) knowledgeable and experienced with Agile development?	Plan	PM leadership is fundamental to align enterprise, program, and development objectives and to ensure the program is conducted in an Agile fashion.
Is there a clear business owner who is dedicated to the effort?	Plan	An authoritative and available business owner is necessary to ensure alignment of the effort with business objectives.
Does the acquisition team have the appropriate skills (i.e., Agile trained, cross-functional, critical domain knowledge)?	Plan	The acquisition team must be knowledgeable of Agile methods as well as domain knowledge to effectively execute an Agile acquisition.
Are the core team members empowered to make decisions on behalf of their communities?	Plan	Empowerment is implicit in the Agile values and principles; it is necessary to achieve the Agile goals of flexibility and rapid development pace.
Is the acquisition team dedicated to the effort?	Plan	Dedicated teams are strongly encouraged to promote productivity and avoid the inefficiencies of context switching between tasks, lapses in collaboration, and impeded communications.
Is the team effectively sized for Agile?	Plan	Agile methods suggest optimal team size based on research and ways to scale team organization for larger efforts.
Is the team co-located?	Plan	To promote communication and collaboration, the Agile ideal is for teams to be co-located. Recognizing that this is not always possible, technical means should be employed to promote coordination.

Are the necessary tools/resources available to the team?	Plan	Especially for new implementations of Agile, it is important to provide suitable Agile training to the team and stakeholders. The Agile implementation may be facilitated by the use of Agile-oriented tools for managing backlogs and measuring progress in an Agile way.
Does the project have an experienced Agile coach/mentor?	Plan	Especially when Agile is being introduced and adapted to a new environment, it is important to have an experienced and trusted advisor to whom the team can turn for guidance and advice.
OPERATIONAL/TECHNICAL ASPECTS		
Does the program involve a new investment or capability acquisition with substantial software development involvement?	Factor	Although Agile principles and practices may have broader application, the origin of Agile and motivation for its evolution have been to promote more effective software development.
Does the capability involved have direct business value?	Factor	The Agile principle of satisfying the customer and practices of prioritizing requirements and managing backlogs are significantly impeded without clarity about the business value involved.
Does the capability involved represent a critical function or potential operational bottleneck?	Factor	Critical functions and potential bottlenecks suggest a combination of time criticality and functional completeness that are not consistent with the Agile expectation of scope flexibility.
Is the prospective program "rightsized" for Agile application (e.g., ACAT 4 or 5) given the current maturity of Agile use in the organization?	Factor	Although approaches exist for scaling Agile for application across the enterprise, greater experience exists with smaller efforts, and smaller efforts are more adaptable as the organization matures its Agile practices.
How subject to change are the capabilities created by this effort?	Factor	A fundamental value of Agile is adaptability and responsiveness to change. If change isn't a property of the environment, then the utility of Agile is diminished.
Does the program need or benefit from rapid and incremental delivery of operational capabilities?	Factor	Agile emphasizes the rapid delivery of value at an established development cadence. If the operational environment constrains the ability to change operational capabilities frequently, then the value of Agile will be diminished.
Is the program sensitive to user acceptance and operational suitability issues that can benefit from early and frequent user feedback?	Factor	Agile promotes frequent exposure of development products to users for feedback and acceptance, promoting operational suitability. If the capabilities involved don't have the sensitivity (e.g., embedded functions with little/no user interface), then the value of Agile will be diminished.
Does the program offer sufficient flexibility to define source selection and contract content that enable/promote Agile development?	Factor	Agile value is undermined if the developer is unwilling or unable to collaborate in the Agile process, or if the contract disincentivizes an Agile approach (e.g., defines a single deliverable, focuses on documentation).

Is there a target architecture (e.g., stable infrastructure and standard interfaces) that would inform software design and development?	Factor	Agile development is greatly facilitated by leveraging existing assets, allowing development to focus on features of program unique value and reducing planning involved.
Is the program providing infrastructure on which other components will rely?	Factor	Agile promotes adherence to cost and schedule, but with scope flexibility offering the degree of freedom needed for adjustment. If other projects are dependent on specific functionality being provided by an Agile project, scope variability may not be available and the Agile development could be over-constrained.
Can the requirements be prioritized?	Plan	Agile is based on evolving requirements, having the flexibility to trade off requirements of lesser importance to adhere to cost and schedule, and pursuing the highest value first. Satisfaction of these objectives involves effectively prioritizing requirements accounting for value, urgency, and dependencies.
Does the acquisition/development have a fixed timescale?	Plan	The Agile emphasis on established cadence and schedule adherence aligns well with a need to meet a fixed timescale. Agile assumes that scope flexibility exists as a trade-off to meet schedule, understanding that effective prioritization ensures satisfaction of critical functionality and performance.
Does the program offer sufficient flexibility to trade off lower priority functions and features for schedule adherence and resource constraints?	Plan	Agile methodology depends on scope flexibility. To promote achieving the greatest value, an Agile development needs a prioritized backlog that doesn't demand completion. Situations may exist where this isn't possible (e.g., the entire development effort is the minimum viable product).

2.3 Example Agile Lifecycle Alternative

Applying Agile principles to the AMS requires tailoring certain acquisition activities and artifacts irrespective of the development lifecycle strategy (e.g., the Waterfall or Agile development practices). The following sections present an illustrative Agile lifecycle model and associated artifacts. In this Agile lifecycle model example, shown in Figure 2 2 and Figure 2 3, the current AMS process has been tailored for a new, software-intensive system. This Agile Acquisition model focuses on the high-valued activities and artifacts (e.g., items necessary to make an informed decision for acquisition), as shown in Figure 2 4, from defining the mission need to establishing an acquisition strategy and business case, and finally to articulating the implementation strategy and roadmap. The primary assumption is that there is flexibility in scope and dedicated, trained Agile individuals to execute this Agile Acquisition. It must be stressed that the presented example is only a draft, and there is a need to develop concurrence on an Agile lifecycle adapted to the FAA environment.

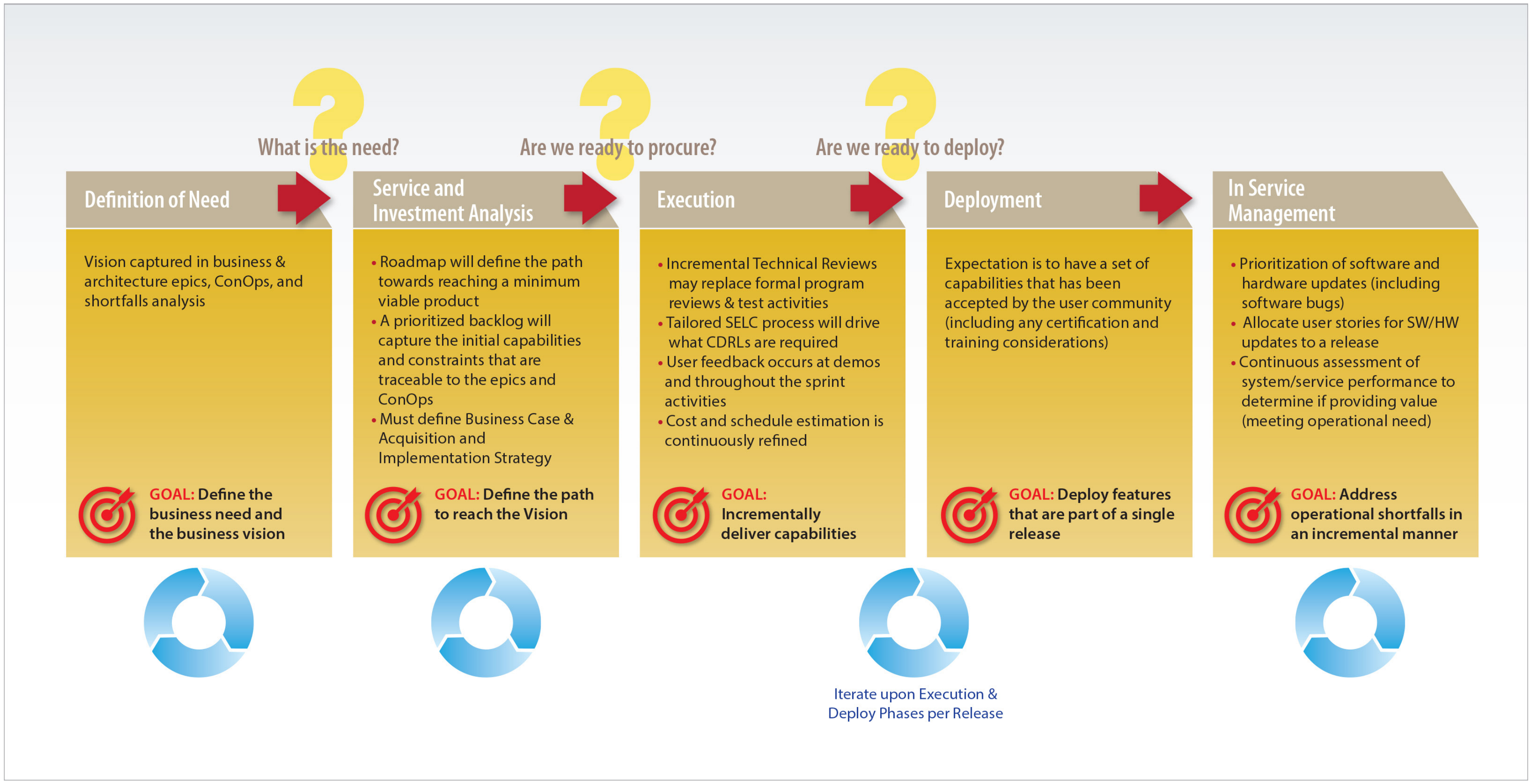
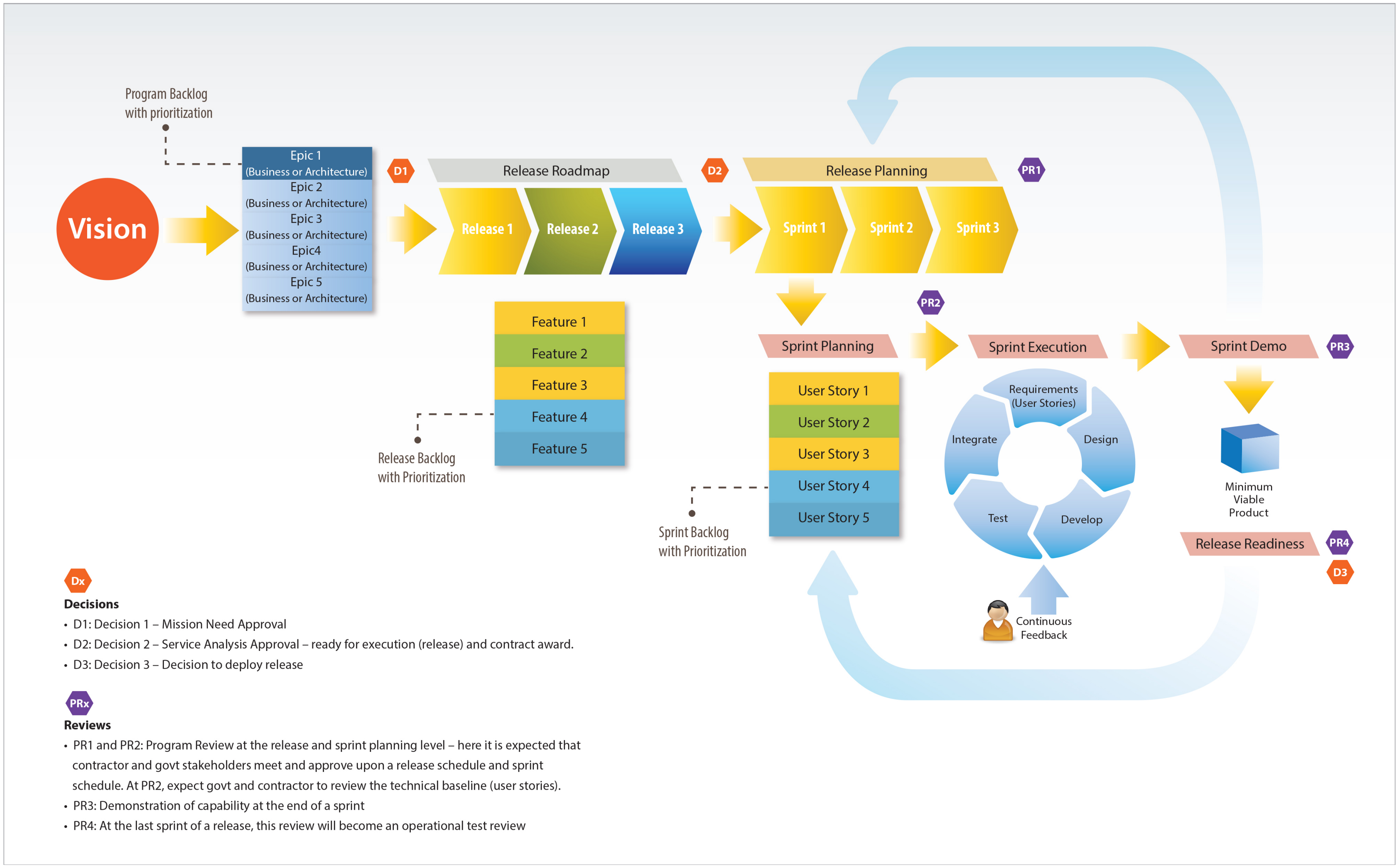

What is the need?
Are we ready to procure?
Are we ready to deploy?
Definition of Need
Vision captured in business & architecture epics, ConOps, and shortfalls analysis
GOAL: Define the business need and the business vision
Service and Investment Analysis
• Roadmap will define the path towards reaching a minimum viable product
• A prioritized backlog will capture the initial capabilities and constraints that are traceable to the epics and ConOps
• Must define Business Case & Acquisition and Implementation Strategy
GOAL: Define the path to reach the Vision
Execution
• Incremental Technical Reviews may replace formal program reviews & test activities
• Tailored SELC process will drive what CDRLs are required
• User feedback occurs at demos and throughout the sprint activities
• Cost and schedule estimation is continuously refined
GOAL: Incrementally deliver capabilities
Deployment
Expectation is to have a set of capabilities that has been accepted by the user community (including any certification and training considerations)
GOAL: Deploy features that are part of a single release
In Service Management
• Prioritization of software and hardware updates (including software bugs)
• Allocate user stories for SW/HW updates to a release
• Continuous assessment of system/service performance to determine if providing value (meeting operational need)
GOAL: Address operational shortfalls in an incremental manner
Iterate upon Execution & Deploy Phases per Release

Figure 2-2 Agile Lifecycle Model



Decisions

- D1: Decision 1 – Mission Need Approval
- D2: Decision 2 – Service Analysis Approval – ready for execution (release) and contract award.
- D3: Decision 3 – Decision to deploy release

Reviews

- PR1 and PR2: Program Review at the release and sprint planning level – here it is expected that contractor and govt stakeholders meet and approve upon a release schedule and sprint schedule. At PR2, expect govt and contractor to review the technical baseline (user stories).
- PR3: Demonstration of capability at the end of a sprint
- PR4: At the last sprint of a release, this review will become an operational test review

Figure 2-3 Agile Acquisition Lifecycle Model Detailed Process

2.3.1 Define Need

During the *Define Need* phase, the FAA is responsible for defining a vision that captures the mission need and an enterprise understanding of the business and architecture capabilities. Aspects of Section 3.2, Agile Planning, and Section 3.9, Enterprise Architecture—Transition to Agile, are relevant during this phase, as it includes defining a vision and roadmap. The operational functions and environment are captured in business and architecture epics, which will feed into the creation of a release roadmap, which is part of the next phase (see Figure 2 3 for details on the relationship among vision, epics, and release roadmap). As a solution has not been explicitly defined or selected, the epics should represent what the desired operational end state should be in a moment in time, devoid of any design specifications. The *Define Need* phase is an iterative process, where feedback loops are used to continuously refine the program backlog, capturing new or changed enterprise (agency) priorities based on input from the user community. Articulating the mission need and desired operational functions in the form of a vision document, a program backlog, and program roadmap is critical to passing this first milestone, as the FAA's decision makers (e.g., JRC) must have evidence of an operational need.

2.3.2 Service and Investment Analysis

The next phase, *Service and Investment Analysis*, focuses on establishing a program that will execute the vision. The business and architecture epics, initially captured in the program backlog, will be decomposed into a set of features, both functional and non-functional, and allocated to a release backlog (see Section 3.3 for more information on Agile requirements). These features are used to guide the development of an acquisition and implementation strategy, as they represent the desired objectives/outcomes on which a vendor will bid and deploy in a set timeframe. Prior to any further refinement of capabilities, the vision, epics, and features will need to be evaluated against the enterprise architecture to ensure that there are no duplicate existing or planned efforts and/or features. A Product and Release roadmap captures the set of features to be included in a Release and the schedule of the Release (see Section 3.2 for more information). Figure 2 3 highlights in yellow those high-priority features that represent a minimum viable product (set of features that adds value from the user community's perspective), which will be included in the first Release. Special considerations such as user training must be accounted for when developing the Release schedule.

The acquisition strategy includes development of a business case, which includes a cost/benefit analysis and budget assessment (see Section 3.4 for information on cost estimation), and a contracting strategy. To support frequent delivery of usable capabilities, the contract must be tailored appropriately to focus on the activities and required deliverables (Contract Data Requirements Lists) that provide high value (see Section 3.5 for information on Agile contracts). For example, the Request for Proposal may include guidance on the expectations for incremental development processes, with more frequent program reviews and specific Agile metrics for measuring performance from a program management perspective. The FAA has a large role in conducting these activities prior to any development work. The expectation at the end of this phase is for the FAA to have set up and identified the necessary resources to support an Agile development process, including identifying a clear roadmap of features and FAA personnel who will serve as a user community representative (Product Owner). Facilitation mechanisms to support continuous feedback and engagement with the Development Team, systems engineering team (including specialty engineering), and the Program Office are established. The following artifacts are developed as required to proceed into the next phase and allow further development work with the contract awardee:

- Release roadmap that captures the scheduled development cycle and the release backlog
- Acquisition strategy, including the ACAT level and contract strategy
- Implementation strategy
- Business case to highlight the cost/benefit analysis and budget assessment.

2.3.3 Execution and Deployment

The *Execution and Deployment* phases focus on the developer's efforts and collaboration with the FAA. Together, the FAA and the developer will form an Integrated Delivery Team, which includes specialized personnel in systems engineering, security, T&E, program management, and contracts, to ensure the successful delivery of capabilities. From a requirements and planning perspective, the developer will begin to define Sprints and user stories (as part of the Sprint backlog) that decompose the release plans and features from the release backlog. From Figure 2 3, three Sprints will address features 1 and 3 as part of the first release, where the Development Team will design, develop, integrate, and test the user stories assigned per Sprint. User stories that cannot be addressed or fulfilled by the end of the Sprint will be reassessed for importance and go into the Sprint backlog for a future Sprint, if deemed a priority. In addition, the release backlog should be re-evaluated for impact of a missed user story. Changes to the backlog should be also reflected in the contract, where the program manager and the contracting officer (CO) should continuously collaborate in defining the contract scope as necessary. The ability to flex on the scope of the work at a Sprint level is for risk reduction purposes, as the Development Team is able to make more informed design decisions. A Test and Evaluation Master Plan will capture the resources needed for test and integrations, and is expected to evolve through the execution of each Sprint (see Section 3.6 for information on T&E).

There are a number of program reviews (milestones) to ensure alignment of expectations between the user community, Program Office, and developer. For example, in Figure 2 3, there is a review milestone, the purple diamond marked PR2, after Sprint planning, where the FAA may sign off on the developer's plans for Sprint execution, specifying the user stories that will be addressed and mechanisms to support integration and test (ask if build plans are executable). The frequency and duration of program reviews, including the entrance and exit criteria, may be based on the complexity and needs of the Program Office. While there is no "right number" of reviews, the Program Office should determine the number of reviews that would provide the most value given a schedule and cost constraint in a more collaborative, incremental development process, and may consider leveraging certain Agile events (e.g., Sprint demonstration) as part of the review. For example, the requirement for a Preliminary Design Review may be fulfilled through a number of smaller reviews, where features are demonstrated to the user community, and a number of required and valuable systems engineering artifacts, including a user stories (requirements-like) document and system allocation baseline for that subset of features, are completed. Sprint demonstration occurs at the end of the Sprint and is a formal activity for the entire program to evaluate the completed set of capabilities and provide feedback, which may become incorporated into the Sprint, Release, and program backlog. Subsequent successful Sprint demonstrations (e.g., number of Sprints required to meet the Release goal) will culminate in the next decision point, Release Readiness, as indicated by the red diamond marked D3 in Figure 2 3.

The decision milestone at the end of the Execution phase verifies and validates that the feature(s) provides business value (addresses the users' needs) and is fully tested and interoperable. Successful completion of this milestone, D3, indicates that the FAA has a completed, working set of feature(s) that is ready to be deployed to the field (i.e., features are operationally effective and suitable). The Deployment phase specifically looks at introducing new solutions in the operational environment for the users, and includes training users and system maintainers and installing hardware and software. The induction of a new operational solution may highlight certain defects that will need to be addressed for full site integration. These defects must be captured in the backlog, prioritized based on input from the field user community, and addressed in incremental Releases.

2.3.4 Support and Maintenance

Following the Deployment phase when the solution is in service, the Support and Maintenance phase leverages continuous user feedback, where the Development Team identifies preventive and corrective maintenance measures for a deployed solution (system or service) [7]. The Development Team composition may change during the Support and Maintenance phase; regardless of the change, the Development Team must have the same goal in mind as in previous phases of the Agile acquisition process, ensuring valuable and working capabilities. The FAA should continuously assess the system/service's performance (e.g., data on feature usage) and its ability to address the mission need throughout the system/service's lifetime. For example, the retrospective may identify changes in the operating environment, requirements, and/or interfaces, and safety and/or security issues (see Section 3.7 for more information on Agile practices related to Deployment and Sustainment). Agile processes provide a way to continuously address the risks of a program, where the user community (including site users and operators), Product Owner, Development Team, and the program manager are all involved in determining allocation of resources and in assessing the operations. This collaboration promotes the Agile principle of delivering fixes and updates in a more predictable and incremental manner, where users' needs are satisfied through the delivery of a working solution.

2.3.5 Agile Acquisition Artifacts

Figure 2 4 depicts the artifacts for each phase of the lifecycle shown in Figure 2 3, where those highlighted in green are highly valuable. The figure is not a complete representation of all possible artifacts and could be expanded to consider any speciality engineering artifacts, including safety and security-specific artifacts, such as an Operational Hazards Analysis. There is a relationship (trace) among these artifacts, where the decomposition of the vision and epics manifests itself as user stories in the Sprint backlog and as other systems engineering artifacts, including the Test and Evaluation Master Plan (TEMP) and any security characterization documents. It is expected that these artifacts will evolve through the lifecycle, increasing in maturity and being updated to reflect changes. Thus, it is important to understand the dependencies in order to assess the impact of a change in the environment and maintain the traceability to the main vision.

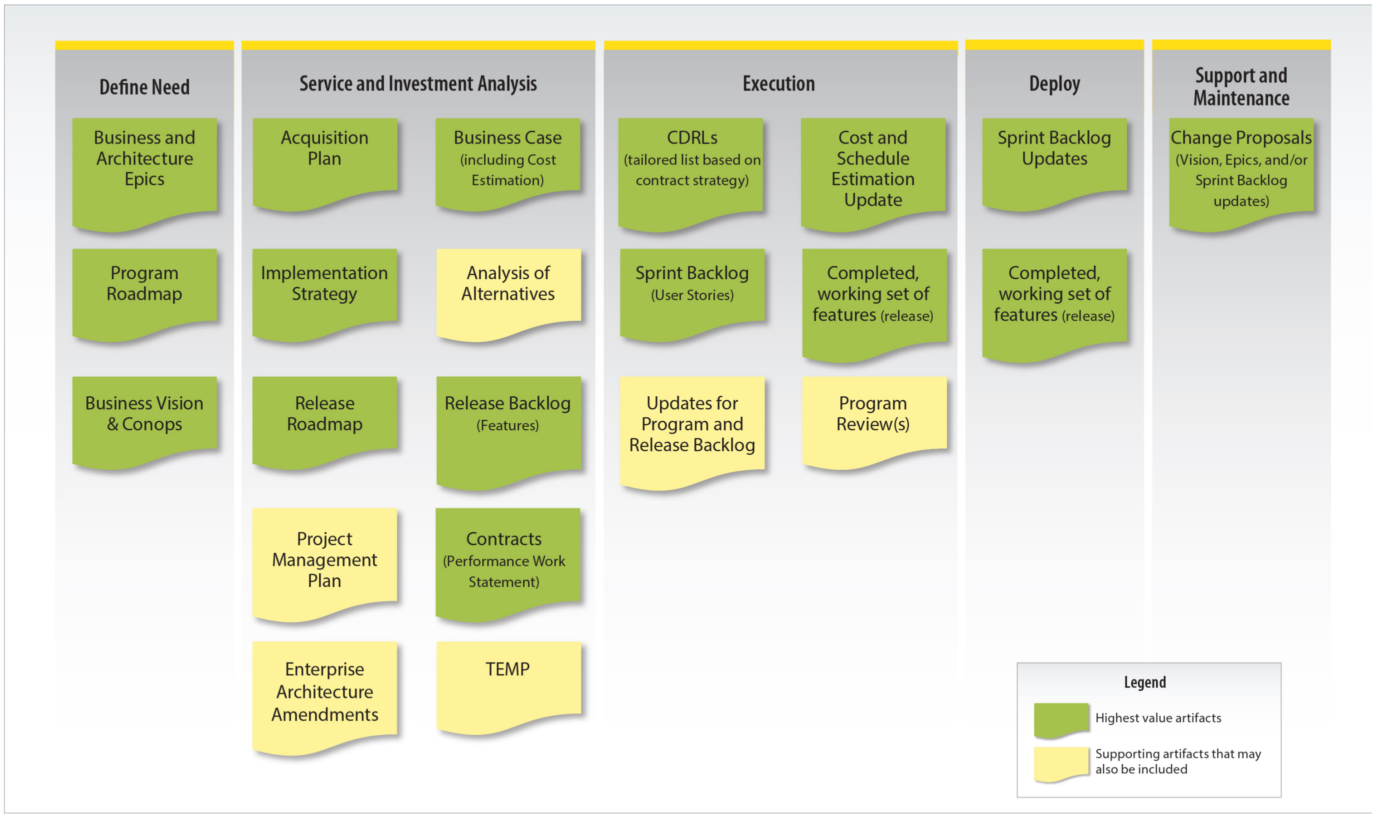

Define Need
Business and Architecture Epics
Program Roadmap
Business Vision & Conops
Service and Investment Analysis
Acquisition Plan
Business Case (including Cost Estimation)
Implementation Strategy
Analysis of Alternatives
Release Roadmap
Release Backlog (Features)
Project Management Plan
Contracts (Performance Work Statement)
Enterprise Architecture Amendments
TEMP
Execution
CDRLs (tailored list based on contract strategy)
Cost and Schedule Estimation Update
Sprint Backlog (User Stories)
Completed, working set of features (release)
Updates for Program and Release Backlog
Program Review(s)
Deploy
Sprint Backlog Updates
Completed, working set of features (release)
Support and Maintenance
Change Proposals (Vision, Epics, and/or Sprint Backlog updates)
Legend
Highest value artifacts
Supporting artifacts that may also be included

Figure 2-4 Agile Lifecycle Model Required Artifacts

APPLYING AGILE WITHIN THE ACQUISITION MANAGEMENT SYSTEM

3

3.1 Agile Culture and Organization

At the heart of an Agile Acquisition approach are the people employed to conduct the roles and responsibilities associated with acquiring, engineering, and developing capabilities. Defining and understanding the Agile team concept is key to the success of the Agile approach. Agile software development methodologies place a strong emphasis on prescribed roles for team members, aligning each member's technical expertise with the technical development needs. When extrapolated to acquisition, the emphasis on roles is the same. A clear identification of the roles and responsibilities associated with the acquisition, systems engineering, program management, and development organizations provides the means for a program to successfully field capabilities in alignment with identified priorities.

The key Agile principles associated with culture and organization are as follows:

- Adopt the Agile culture
- Integrate Agile knowledge
- Define Agile roles and responsibilities
- Develop organizational agility
- Scale Agile practices as necessary.

3.1.1 Adopting the Agile Culture

One of the most important elements to consider in moving to an Agile paradigm is institutional culture. The Agile culture is a departure from the culture of institutions that have evolved in conjunction with traditional approaches to acquisition, engineering, and development. An important aspect of introducing Agile is to work with the existing community in a systematic way in order to promote adoption to the new paradigm. Organizational change and design strategies are important elements to consider in this process.

The following practices can help to ensure that organizations and programs successfully adopt an Agile culture:

- Identify a senior champion for the Agile approach. A senior champion with authority and responsibility for the program can bolster sponsorship of the effort across organizations, and resolve issues when necessary. The importance of this role cannot be overstated [12].
- Develop a culture of trust. Agile relies on close collaboration and empowerment of teams across organizational boundaries, both internally to the FAA and externally with the contractor(s). A culture of trust that spans these organizations is a critical aspect of creating an environment in which the

Practices for Adopting the Agile Culture

- **Identify a senior champion for the Agile approach**
- **Develop a culture of trust**
- **Develop a strategy for culture change**
- **Align commitment across organizations**
- **Utilize a dedicated acquisition team**

engineering and development work can thrive. An important consideration in this regard is to help shift the expectations associated with the "big bang" Waterfall development approach to one of continuous capability evolution [7], where program managers may trust the Agile approach and team's judgement.

- Develop a strategy for culture change. It is useful for a program adopting an Agile approach to identify each of the organizations impacted by the program, and ensure that there is a plan for educating others on the new approach and any potential changes to roles and responsibilities.
- Align commitment across organizations. Each of the organizations impacted by the Agile approach must be committed to the new model. An Agile approach may prescribe changes to existing roles and responsibilities. These issues should be dealt with openly in a manner that facilitates learning about the process and emphasizing the value of the approach. A senior champion can be a valuable resource in developing organizational alignment.
- Utilize a dedicated Acquisition Team. Whereas the necessity of a dedicated Development Team is well established in Agile guidance, it is important to note that the role of the Acquisition Team is equally critical to the success of an Agile program. In early phases of the acquisition process, a dedicated Acquisition Team helps to establish and ensure the continuity of the vision and goals for the program across organizations. During implementation, the Acquisition Team works with the developer on a daily basis to serve Product Owner roles and responsibilities, manage evolving requirements, and serve as an interface to organizations beyond the program boundary [7].

3.1.2 Integrating Agile Knowledge

A clear understanding of the Agile mind-set and methodology is necessary to create the circumstances for success with Agile Acquisition. Knowledge among Development Team members is critical for Agile software development. Also, the acquisition and systems engineering organizations need to have the knowledge necessary to facilitate acquiring, engineering, and managing using Agile methods. A consistent level of knowledge across communities also provides for a common vocabulary when attempting to plan and execute an Agile Acquisition.

Agile knowledge can be integrated through a variety of techniques. The following practices can help programs to assimilate knowledge:

- To the extent feasible, it is recommended that programs recruit personnel with Agile expertise who have worked within other FAA programs. If these skills are unavailable, personnel with Agile backgrounds from other agencies or commercial organizations can also be suitable in addressing this need.

Practices for Integrating Agile Knowledge

- **Recruit personnel with Agile expertise who have worked within other FAA programs**
- **Leverage an Agile coach to ensure that programs are proceeding down an appropriate path**
- **Utilize training to enable a successful Agile Acquisition**

- Leveraging an Agile coach can be a valuable means to ensure that programs are proceeding down an Agile path. Coaches are experts in Agile methods, and should have several years of experience with Agile methods across different roles and organizations. Coaches can also provide on-the-job training to personnel, and help to identify and address gaps in Agile process or implementation.
- Training is an important component of enabling a successful Agile Acquisition. At the software development level, it is important that each team member be familiar with the Agile methodology. Training within the program management organization is another key enabler of success. In general, given the disparate level of Agile knowledge within the government community, training should be employed to provide a common frame of reference for the program team and serve as an enabler for Agile Acquisition.

3.1.3 Defining Agile Roles and Responsibilities

Within the commercial arena, the roles and responsibilities associated with Agile software development are fairly well defined. The Agile team typically comprises the Product Owner, the Development Team, and the Scrum Master. The key to successful Agile execution for the FAA is to translate these roles to a government environment. Within the FAA, three perspectives need to be considered for roles and responsibilities: the Agile team level, the program level, and the enterprise level.

Practices for Defining Agile Roles and Responsibilities

- **Product Owners maximize the value of the work conducted by the Development Team**
- **Development Teams conduct the work necessary to deliver a potentially releasable increment of functionality at the end of each Sprint**
- **Scrum Masters maintain the structure, and the scrum rules of operation**
- **Program managers ensure the program is structured and can function in an agile fashion**
- **Lead engineers ensure that the system is developed in a manner that aligns with the FAA's evolving enterprise architecture and technology standards**

While several of the roles at the program and enterprise levels within the FAA are well known and understood, it is important to distinguish between typical responsibilities and those needed for programs adopting Agile methods.

Figure 3 1 identifies key roles necessary for effective execution of an Agile Acquisition at different levels. At the core is the Agile team level, which emphasizes the conduct of development work and delivery of functional increments on the program's cadence. Beyond the Agile team is the Program level, which focuses on the planning and execution of work from inception to completion. The Enterprise level serves various functions, which apply to the full gamut of programs within the FAA enterprise.

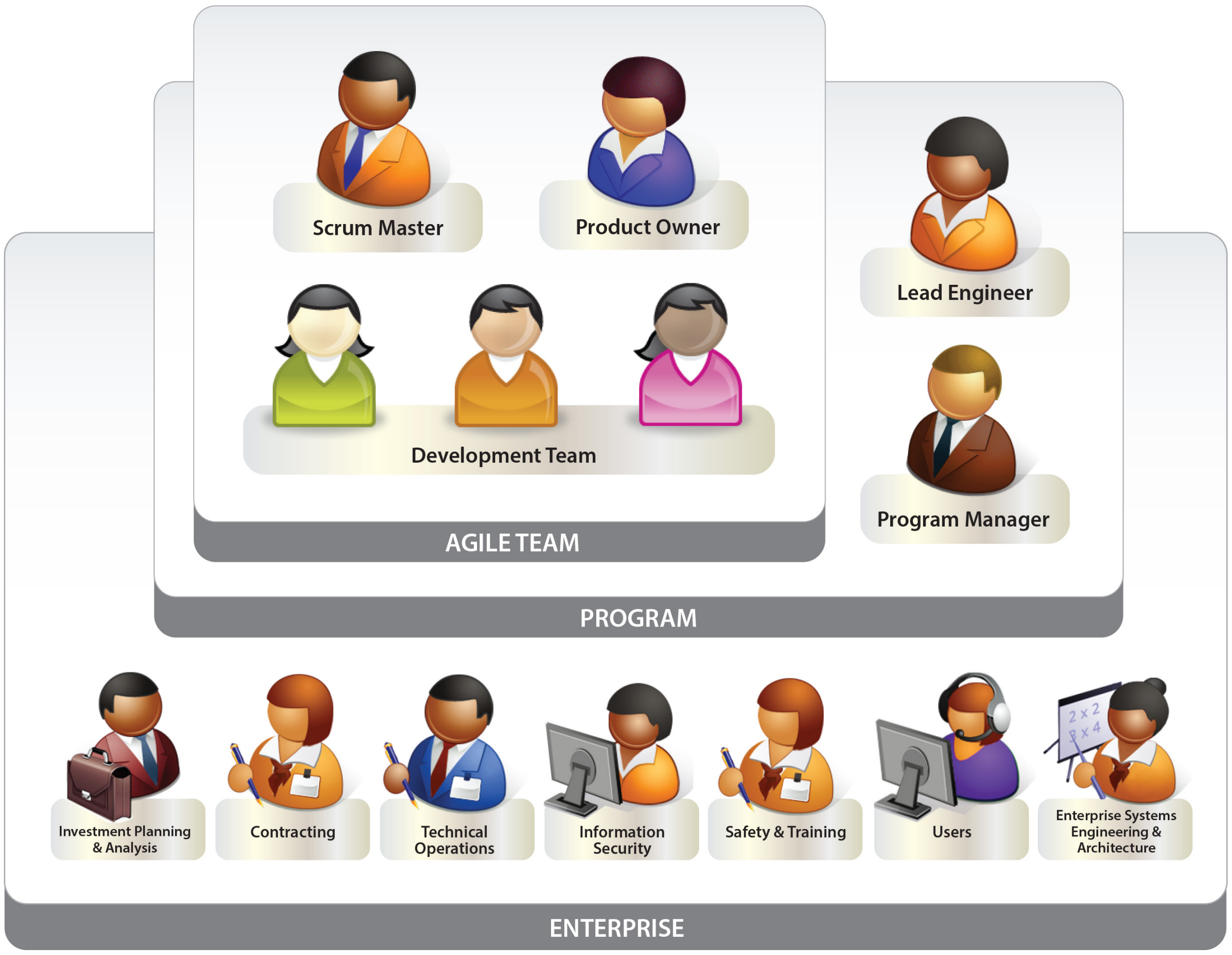


Figure 3-1 Key Roles for Effective Execution of Agile Acquisition in the Enterprise

The following outlines key roles and responsibilities for the Agile team within the FAA [13]:

- The **Product Owner** is responsible for maximizing the value of the work conducted by the Development Team. Within the FAA, the Product Owner is the authorized representative of the user community, interfaces with interacting programs, manages the prioritization of the backlog, and regularly gives technical direction to the Development Team to provide the operational perspective. The Product Owner also approves and accepts stories and provides acceptance criteria and the definition of "done." The Product Owner should be a government person. Given the FAA's organizational structure and the complexity of FAA programs, this role may be subdivided into technical and operational leads (e.g., to obtain field representation of a user perspective). However, lines of authority need to be clearly identified.
- The **Development Team** conducts the development work necessary to deliver a potentially releasable increment of functionality at the end of each Sprint. The Development Team consists of the full gamut of skills needed to deliver the release. This list can include systems engineers, software developers, testers, integration engineers, architects, security engineers, data specialists, and IT operations specialists.
- The **Scrum Master** is responsible for maintaining the structure, and the Scrum rules of operation. The Scrum Master provides a "servant-leader" function to the Product Owner and Development Team. Typical responsibilities include helping the Product Owner with backlog management, removing barriers to the Development Team's progress, and serving as a coach to ensure that the Development Team maximizes value delivery.

Practices for Defining Agile Roles and Responsibilities (2)

- **Identify an organization to work with the program to estimate costs and benefits for each increment**
- **The contracting organization executes contracts that solicit a mix of Agile and domain specific subject matter expertise**
- **The testing, information security, and safety & training organizations work on the program's cadence to perform their mission**
- **Users provide input and feedback throughout the program's lifecycle**
- **The enterprise systems and architecture organization aligns the program to enterprise technologies and standards to ensure interoperability with other programs**

The following list identifies key roles and responsibilities at the program level within the FAA:

- The **program manager** is responsible for overseeing the planning and execution of the program and the conduct of the government Integrated Product Team (IPT). The program manager works within and outside the program to ensure that it is structured such that it can function in an Agile fashion, especially with respect to the delivery of prioritized capabilities on cadence. The program manager plays a key interface role between the enterprise and team levels, and builds awareness across communities to help enable agility. An important responsibility of the program manager is to ensure that the Agile team has an FAA Product Owner who is fully dedicated to the development effort. From an enterprise perspective, significant responsibilities include working with the business owners to identify capabilities and acceptance for releases, working with the investment planning and analysis organization for increment-level budget planning, and working with the contracting organization to define the appropriate contract type and solicit requisite Agile software development expertise.
- The **lead engineer** works closely with the program manager to oversee the technical aspects of the program. On some programs, the lead engineer role may be fulfilled by the program manager. Like the program manager, the lead engineer plays an important interface role between the enterprise and team levels. The lead engineer works with architects and systems engineers to ensure that the system is developed in a manner that aligns with FAA's evolving enterprise architecture and technology standards. At an enterprise level, key responsibilities include working with other programs on integration and technology-related issues as well as working with the technical operations, information security, and safety and training organizations as appropriate to ensure that capabilities can be delivered on the program's cadence.

The following list identifies key organizational roles and responsibilities at the enterprise level within the FAA:

- Identify an organization to work with the program on estimating costs and benefits for each increment.
- The contracting organization helps to select, define, and execute contracts that solicit for the appropriate mix of both Agile and FAA domain-specific subject matter expertise (e.g., Air Traffic Control system specific). The contracting organization is responsible for certifying that work is complete, including oversight, accepting products and deliverables, and ensuring that invoices are paid. A designated contracting representative may be a dedicated member of the IPT.
- The testing organization works with the program on verification and validation of capabilities on the program's cadence.
- The information security organization works to authorize, assess, and monitor FAA capabilities for information security risks.
- The safety and training organization assesses the capabilities for potential impacts to safety and works with the program to ensure that products are developed at an appropriate level of rigor. The safety and training organization is a key partner in ensuring that users are trained and certified in alignment with the program's cadence.
- Users provide input and feedback throughout the program's lifecycle. Input can be provided during demonstrations, training sessions, and forums held for the explicit purpose of garnering user

input. Typically, it is envisioned that the product owner will serve as the interface to users on the program's behalf.

- The enterprise systems engineering and architecture organization plays an important role in aligning the program with enterprise technologies and standards to ensure interoperability with other programs. Typically, it is envisioned that the lead engineer will play a key role in working with the enterprise systems engineering and architecture organization throughout the program's lifecycle.

3.1.4 Developing Organizational Agility

Practices for Developing Organizational Agility

- **Use fully dedicated resources to the extent possible**
- **Adopt the prescribed roles and responsibilities of the Agile team**
- **Co-locate the Agile team as much as possible; effectively integrate distributed members if necessary**
- **Effectively integrate the IPT and developer teams**

Organizational structure and behavior are important elements to consider within the paradigm of Agile Acquisition. The organizational structure can either facilitate or hinder the delivery of an Agile program. Structural considerations are especially important within the program team delivering capability. However, attention also needs to be given to external organizational elements interacting with the program team.

Key practices related to developing organizational agility include the following:

- Use fully dedicated resources to the extent possible. Research has demonstrated that fully dedicated team members are significantly more productive than those split across multiple tasks and projects [14].
- Adopt the prescribed roles and responsibilities of the Agile team. The structure of the Agile team is a time-tested key enabler of success within the Agile development model. Within the FAA, roles such as systems engineers and architects do need to be considered where necessary. To keep coordination overhead to a minimum, however, it is recommended that Agile teams refrain from adding roles that do not directly contribute to the development effort.
- Co-locate the Agile team as much as possible, and effectively integrate distributed team members if necessary. Successful Agile teams are co-located and are able to have face-to-face conversations whenever necessary [2]. However, FAA programs that leverage multiple vendors may make co-location across corporate boundaries infeasible. In such instances, it is recommended that each vendor team be co-located. Where necessary, SoS sessions can leverage videoconference technologies for effective communication.
- Effectively integrate the IPT and developer(s). Agile practices stress the need for highly involved

government personnel (e.g., Product Owner) who are ideally available on a daily basis to provide input for the Agile team. This may involve both a culture shift and training in new roles and responsibilities. It is important that the IPT and developer evolve the trust and openness needed to lay the groundwork for a successful collaboration.

3.1.5 Scaling Agile Practices as Necessary

Scaling Agile Practices as Necessary

- **Establish separate Agile teams to scale with the size and complexity of the program**
- **Share Scrum Masters and Product Owners across multiple teams where practical**
- **Utilize a Scrum of Scrums model to coordinate across Agile teams**
- **To the extent necessary, separate Agile teams across corporate boundaries**
- **If the size of the program exceeds 150 people, segment the program into multiple efforts**

Agile teams are most effective when the size of the team ranges between five and nine people [15]. Fewer members typically limit the amount of work that can be accomplished (e.g., lack of necessary skill sets), and more can create a coordination burden. Given the complexity of FAA programs, the development effort for a program may not be effectively undertaken by a team of nine people. Therefore, it is likely that several Agile teams would be needed. This is where scaling Agile practices comes to bear.

The following practices are recommended for scaling the level of Agile activity within a program beyond the team level:

- Establish separate Agile teams to scale with the size and complexity of the program. The number of Agile teams should reflect the development effort commensurate with the program's objectives, budget, and timeline. By being strategic about dividing teams, programs can attempt to minimize the amount of integration needed across teams. Through the process of retrospectives, programs will increasingly learn and improve their understanding of the number of Agile teams required to deliver increments of functionality on cadence.
- Share Scrum Masters and Product Owners across multiple teams, if necessary. Given today's culture at the FAA and within industry, it is unrealistic to assume that Scrum Masters and Product Owners will be dedicated resources available to the Agile team. Sharing these roles across teams provides improved integration across the program and can enhance the effectiveness of the Agile teams.
- Utilize an SoS model to coordinate across Agile teams. An important aspect of scaling entails synchronizing activity across the different Agile teams involved in the development effort. An SoS brings together the Scrum Masters for each of the Agile teams involved in the effort as a means to

solve issues across Agile team boundaries. A similar model should be employed to coordinate the efforts of systems engineers and architects across multiple teams.

Figure 3 2 describes a model in which four Agile teams exist. Their activities are coordinated using an SoS model.

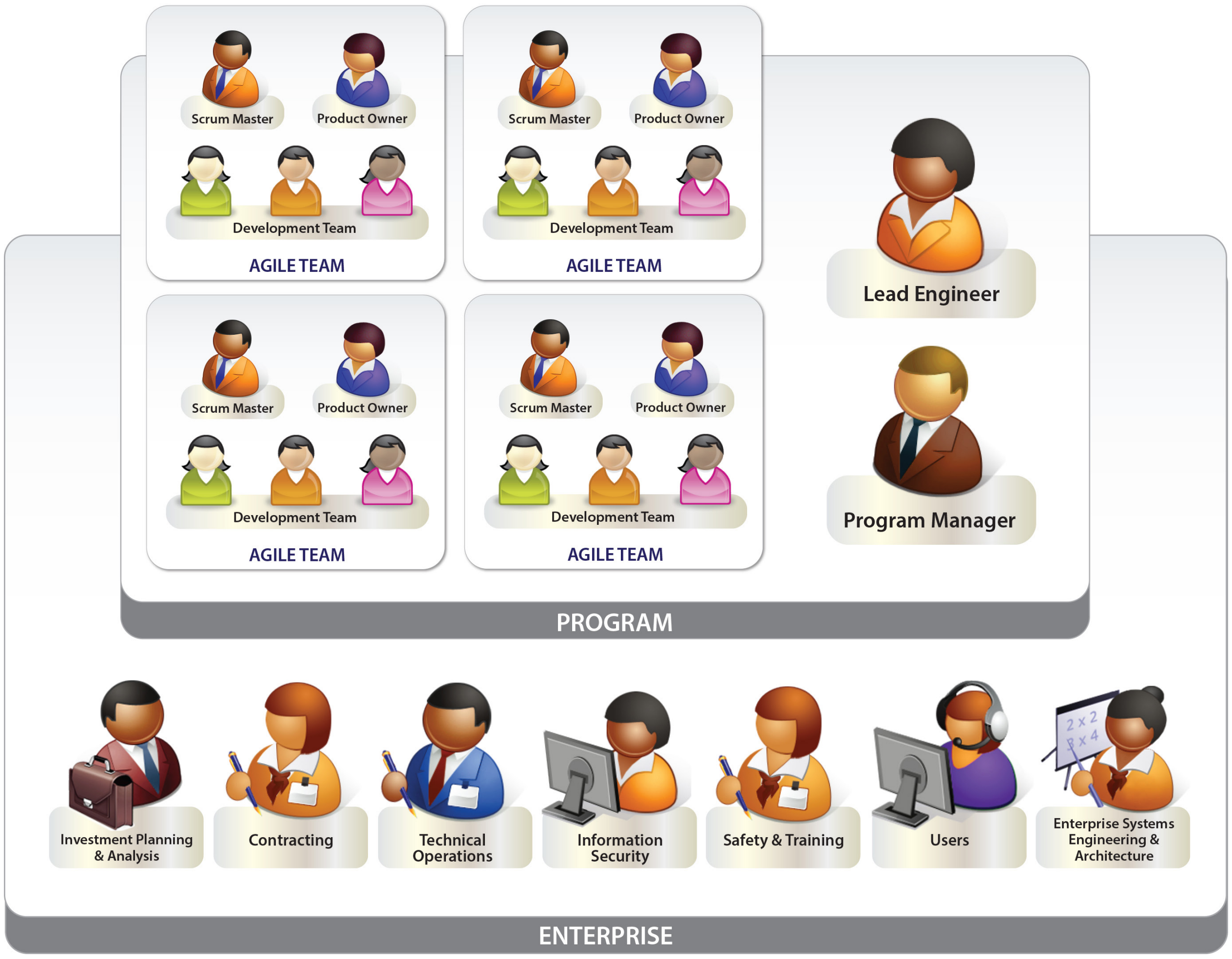


Figure 3-2 A Multi-team Scrum of Scrums Model

- To the extent necessary, separate Agile teams across corporate boundaries. FAA programs may leverage multiple contracts and vendors. Agile teams require a fair amount of flexibility in order to plan and execute their work, and co-location is an important consideration. In addition, interferences external to the project need to be removed to the greatest extent practical. Hence, it is recommended that programs with multiple vendors have vendor-specific Agile teams unless integration across teams can occur at the SoS and other levels.
- If the size of the program starts to exceed 125–150 people, segment the program into multiple efforts. As size and complexity increase, it is important for programs to be mindful of Dunbar's number [16], which suggests that the productivity based on the ability to maintain an effective social network decreases after the program exceeds 150 people.

Figure 3 3 illustrates a means of segmenting a program into multiple efforts. The size and complexity of the program necessitates 150 or more people across the program. To address this challenge, roles at the program level include a program manager, a lead engineer, a deputy program manager, and a deputy lead engineer. These roles coordinate at the program level to synchronize efforts across the segments. As illustrated, each segment leverages multiple Agile teams for the development effort. Those activities are coordinated using an SoS model.

3.1.6 Agile Culture and Organization Challenges

Adopting an Agile paradigm, particularly for early adopter programs, may have challenges in each of the areas described in this section. These include the following:

- Adopting the Agile culture: The FAA's culture generally conforms to accepted methods and practices, which have been utilized by other FAA programs. While Agile practices have gained prominence within the broader government community, the FAA lacks experience in their application. Given this issue, identifying a senior champion for the Agile approach may be challenge. In addition, due to the lack of Agile knowledge and expertise, developing a culture of trust and aligning commitment across organizations may face barriers. To help address the challenge associated with trust and aligning commitment, it is important to develop and pursue organizational design and culture change strategies. An organizational change management plan can be leveraged to help transition FAA organizations into successfully adopting the Agile paradigm. In addition, existing Agile projects are underway within the FAA, and it would be prudent to select a senior champion who has familiarity with or purview over these efforts.
- Integrating Agile knowledge: An important aspect of integrating Agile knowledge is the ability to leverage personnel with Agile expertise on FAA programs. Given the lack of Agile experience within the FAA, identifying candidates with Agile experience is likely to be challenging. Experienced Agile practitioners may be found in other government agencies, and industry. In addition, the FAA may leverage organizations such as General Services Administration's 18F for Agile coaching to guide project teams through the adoption of Agile practices during the project's lifecycle.
- Defining Agile roles and responsibilities: An Agile approach may imply a departure from existing and typical organizational roles and responsibilities. Cultural and organizational issues, including the empowerment of teams, may impede progress. Prioritization is an important consideration in Agile. The need to balance user priorities with those within other parts of FAA's enterprise, including the need to weigh and compare priorities, may be a challenge. Organizational design and culture change strategies, and the organizational change management plan, should account for the issues and challenges associated with transitioning roles and responsibilities where necessary. Experience gained from pilot Agile projects will be invaluable to the FAA gaining expertise with the process of prioritization.
- Developing organizational agility: In the existing culture, government personnel tend to be assigned to multiple efforts; hence, ensuring that the government has a team dedicated to the Agile effort may be a challenge. Given the prevalence of geographically distributed Development Teams, co-location may introduce challenges for developer and government personnel. A dedicated team is a

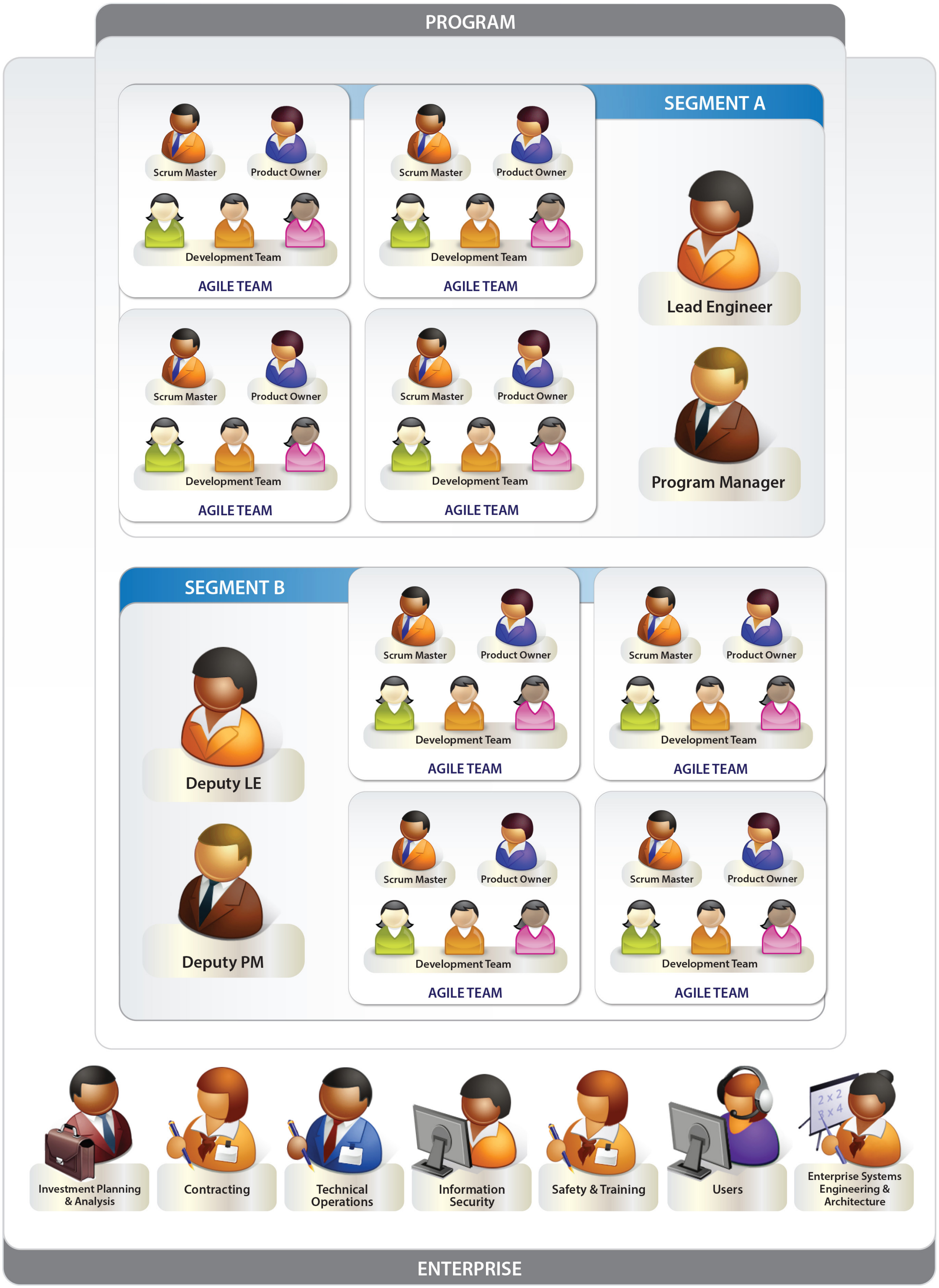


Figure 3-3 Segmenting a Large, Complex Program

key enabler to the success of Agile projects. In the long term, the organizational change management plan can help the FAA transition into this paradigm for Agile projects. In the near term, a compromise may be pursued, such as the ability to have team members dedicated between three and four days a week. If the latter option is pursued, it is important for as many team members to be dedicated to the effort as possible during an overlapping time period to facilitate communication. Where infeasible, the need for co-location may be mitigated by technologies such as audio and video conferencing, instant messaging, and online meetings.

- Scaling Agile practices as necessary: As FAA organizations currently lack experience with Agile practices, scaling them to an appropriate level might prove challenging. Coordinating across Agile teams and corporate boundaries, in addition to appropriately synchronizing the efforts, may also be difficult. In general, it would be beneficial for FAA programs to obtain sufficient Agile experience prior to introducing some of the complexities that scaling practices may bring. If large-scale Agile efforts are pursued in the near term, before the FAA has the opportunity to gain maturity with Agile practices, it is important that Agile coaching and training opportunities be pursued to the greatest extent feasible. One option to consider is to hire a prime contractor with strong credentials in Agile, with the responsibility to hire the subcontractors with the Agile and domain expertise necessary to address the needs of the program. An SoS model and segmentation strategies should be pursued to address the coordination and complexity associated with large-scale efforts.

3.2 Agile Planning

Within the AMS, planning evolves from the strategic considerations of need, alternatives, and required resources, to the tactical planning involved with executing solution development and implementation. Agile approaches must meet the same planning objectives.

The key Agile principles associated with planning are as follows:

- Defining multi-level plans for evolving stakeholder needs
- Refining plans continuously.

3.2.1 Defining Multi-level Plans for Evolving Stakeholder Needs

Practices for Defining Multi-level Plans for **Evolving Stakeholder Needs**

- **Capture the high-level statement of need in the vision**
- **Capture the development schedule (the number of releases needed) to deliver working features incrementally in a product roadmap**
- **Focus Sprint planning on how to achieve the release's objectives**

Agile Acquisition planning activities must be tailored to support a more streamlined Agile Acquisition. Constant coordination among the program, systems engineers, the user community, contracting, T&E, and the Development Team are required to support frequent delivery of capabilities. Considering Agile requirements practices, for example, planning activities include breaking down epics (large user stories spanning multiple development iterations that require decomposition) into incremental releases (smaller implementable user stories) based on features and evaluating the appropriate resource needs.

Industry Agile development practitioners [17] specify five levels of planning involving increased planning precision. Figure 3 4 depicts the Agile planning hierarchy.

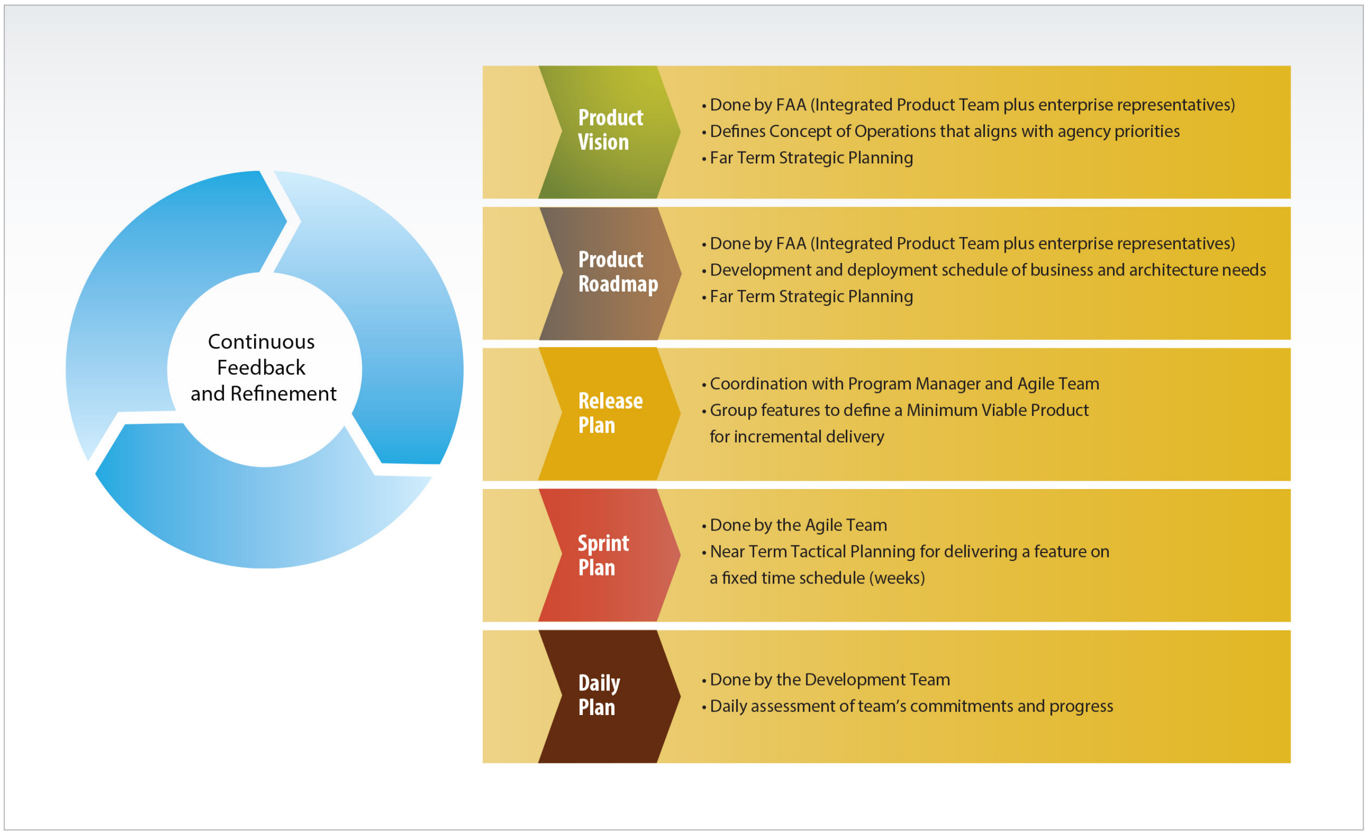

Continuous Feedback and Refinement
Product Vision
• Done by FAA (Integrated Product Team plus enterprise representatives)
• Defines Concept of Operations that aligns with agency priorities
• Far Term Strategic Planning
Product Roadmap
• Done by FAA (Integrated Product Team plus enterprise representatives)
• Development and deployment schedule of business and architecture needs
• Far Term Strategic Planning
Release Plan
• Coordination with Program Manager and Agile Team
• Group features to define a Minimum Viable Product for incremental delivery
Sprint Plan
• Done by the Agile Team
• Near Term Tactical Planning for delivering a feature on a fixed time schedule (weeks)
Daily Plan
• Done by the Development Team
• Daily assessment of team's commitments and progress

Figure 3-4 Agile Planning Stages

For the FAA, this translates into having planning activities at the very beginning of the acquisition with the development of the product vision, including a Concept of Operations, and the product roadmap (schedule). These product outputs are influenced by the agency's funding cycles, enterprise vision, and enterprise architecture roadmap, including any agency priorities, standards, and best practices. Program planning must occur at the Release, Sprint, and daily levels, and consider cost, schedule, and resource constraints. While there are some similarities to the AMS planning process, the differences lie in the level of detail and the planning horizon. For example, the product vision may capture the agency's goals in a five-year plan at a very high operational level, and the roadmap may depict a strategic evolution timeline showing the evolution of the operational environment within those five years.

The product vision and roadmap look into the future (on the scale of years) and prioritize the operational needs and associated epics for development. The release plan is a derivation of the vision and roadmap and is a narrower look ahead, about six months, where specific features are selected to be addressed through Sprints. The planning until this point is more strategic. It focuses on how the program can incrementally deliver value through major releases (as defined through prioritization of features), and usually involves the user community and the program managers. Commitment to these releases occurs at release planning and at the Sprint and daily levels, where Development Teams determine and negotiate the work the team commits to deliver within the Sprint.

Sprint planning is tactical in nature and is based on past accomplishments and the backlog of user stories (description of a desired feature from an end-user perspective). Sprint planning occurs at the beginning of the Sprint and establishes the goals for a particular Sprint cycle (whether it is accomplishing code development for a new capability or determining the baseline subsystem architecture). The goal of Sprint planning is to outline how the Development Team will achieve certain release objectives during a fixed period of time (i.e., the Sprint duration). The daily commitment plan is specific to Daily Scrums, where the Development Team and the Product Owner assess the Sprint plan and the team's abilities (resource allocations and task estimations) and make necessary adaptations based on the team's current status. (The Scrum Training Series [18] provides an overview of the Sprint planning process.) The focus of these daily planning meetings is more specific to individuals on the Development Team and their assigned tasks. At these lower levels of planning, the Development Team and the Product Owner are contributors to the planning activities [19].

3.2.2 Refining Plans Continuously

Practices for Refining Plans Continuously

- **Continuously refine Agile plans**
- **Ensure alignment from vision to Sprint plan**

Regardless of the level, planning activities must consider how to address the users' needs while evaluating the priorities, architecture significance, and risk [20] in order to define the following:

- What needs to be done?
- When does it need to be done?
- Who will perform the work that needs to be done?

As Agile development is based on continuous feedback; the same principle should be applied for Agile program planning. The program manager and Agile Team should continuously assess inputs and changes made at the lower level (Sprint and daily) for impact and capture them in the previous planning level to ensure alignment. For example, changes at the Sprint planning level, including not fully completing a user story, may affect the current release plan. As a result, the program manager and Agile Team should reassess the release plan based on risk and reprioritization. The intent of planning activities is to ensure a common baseline understanding among all players of the Agile Acquisition lifecycle of the program's goals on several different timeframes. It is important to negotiate what is feasible to include in a release (as part of the release plan) and Sprint with what the user community has deemed as of highest value (priority).

3.2.3 Agile Planning Challenges

Agile planning occurs at five different levels, as shown in Figure 3 4, and requires a certain level of dedication to ensure a common comprehension of the agency's and program's plans. Challenges in adopting this Agile planning method include the following:

- **Continuous feedback and refinement at all levels:** Current practices do not rigorously enforce having program managers provide updates into enterprise plans based on changes in their program plans (including baseline scope) on a more frequent basis. Agile planning is dependent on having continuous refinement based on changes at the lowest level (Sprint plan) that is communicated and captured in release plans and product vision and roadmap. The Government Accountability Office (GAO) noted that a challenge with Agile is that "Staff had difficulty committing to more timely and frequent input." Agile planning includes explicit plans and goals for the release and Sprint levels [12]. One way to address this challenge is to have members of the dedicated Agile Team, program manager, and lead engineer fully engaged, providing input, and informed throughout planning. Having planning events at the start of release and Sprint execution encourages the practice of continuously re-evaluating and adjusting the current plans based on the team's feedback.
- **Planning discontinuity due to team disengagement:** Agile planning requires continuous inputs from the user community, Development Team, and program manager throughout the lifecycle. Changes, including personnel changes in roles, may create discontinuity in the Agile planning activities. The GAO noted that a large challenge in employing Agile in the federal environment is "Teams had difficulty collaborating closely" [12]. Collaboration tools, such as video conferencing equipment, may provide opportunities for continuous collaboration and a way to ensure inclusion and accountability among the Agile Team and IPT. In addition, retrospectives may be used to identify challenges in the Agile process, which may be contributing to the lack of engagement.

3.3 Agile Requirements

Requirements engineering captures the "what" and "how well" the acquired solution (e.g., service or system) must perform to address the mission need. The "what" and "how well" represent the functional and non-functional (constraints and performance-related) requirements. This activity starts with the development of a "business need," which through a series of analyses becomes the actual requirements that the developer uses to build the system. Agile practices address the risk of uncertainty through the evolution of requirements, highlighting close collaboration with the user and iterative delivery of functionality. In an Agile context, requirements solicitation and management must change to promote delivery of valuable products in an iterative manner (e.g., magnitude of weeks or months).

The key Agile principles associated with requirements are as follows:

- Satisfy the customer's needs.
- Evolve requirements and reflect regularly.

3.3.1 Satisfying the Customer's Needs

Practices for Satisfying the Customer's Needs

- **Capture functional and non-functional requirements (e.g., security) as user stories**
- **Ensure that user stories are executable by the Development Team**
- **Leverage prototyping activities to solicit requirements**
- **Ensure traceability with the vision**
- **Clearly articulate the acceptance criteria and prioritization levels associated with each user story**

Requirements capture the technical capabilities that will address the operational need. When compared to traditional systems engineering methods, such as the Waterfall model, the manner and method of soliciting and capturing requirements changes in an Agile context. Together, the Product Owner/user community and Development Team, which includes testers, will solicit functional and non-functional requirements (and metadata), which are captured as epics [21], features, and user stories. Having a cross-functional team ensures that all needs are being addressed and captured appropriately. An epic is a large user story that reflects an enterprise initiative. Features are derived from the epics into high-level capabilities that will address the operational need. They are realized in user stories and capture the value of the business need. User stories depart from the traditional requirement format, which states that the "system shall do X" [22], and detail the interactions between a user and the system. User stories capture the desired action to be performed, the user, and the rationale for the desired feature. For non-functional requirements, the user story should reflect the technical function (e.g., undesired consequences that the system should prevent, or how reliable it must be) as it serves as a constraint to the design.

The following provides an example of an epic, feature, and user story, highlighting the relationship where features and user stories specify functionality for a surface movement management tool described in the epic:

- Epic: As an air traffic controller, I need a surface decision support tool to manage surface movement.
- Features related to the epic for surface movement management tool: Surface Events Scheduler and Airport Resource Manager.
- User Story: As a ground controller, I can view the departure gate for a flight in order to manage airport resources.

This generic characterization of requirements promotes a basic common understanding of the business and technical needs (e.g., why a particular function must be performed), while still promoting design flexibility and technical exploration. While user stories only capture the functional interactions and design constraints, the addition of acceptance criteria and definition of done ensures that the Development Team has a complete understanding of the business need [23]. The acceptance criteria may be in the form of usability requirements and performance metrics. Identifying dependencies among user stories is another acceptance criteria attribute that may be included in how a user story is defined. Providing the associations among user stories enables the Development Team to understand the scope of the work and the impact of certain changes. The definition of done focuses on the requisite actions prior to declaring completion of a user story. It includes any test and integration-related activities that must be performed to meet acceptance. In addition, user requirements (stories and features) may also be captured via prototyping activities. Due to the complex nature of certain operational capabilities, prototypes may serve as a means to validate and assess the operational need and to derive specific user stories and features that may have not been initially realized (as captured in the epic and/or vision).

The intent behind this requirements solicitation method is to capture sufficient information to support confidence that the organization is building the right thing without deriving specific technical details in an attempt to "gold plate" or over-specify the design up front. User stories highlight the set of interactions that will add value, and are loosely related to use cases, where use cases may help solicit user stories. As the user community and Product Owner provide inputs to the evolving business and technical needs, requirement solicitation in an Agile context is a continuous activity, where new features and user stories will be created in order to capture the changes—newly identified business needs or modifications—in the environment.

3.3.2 Evolving Requirements and Reflecting Regularly

As change is welcomed in an Agile environment, the backlog houses these functional and non-functional user stories, including any new or changed required functions, the acceptance criteria, and any prioritization levels provided from the stakeholders. A program backlog is forward looking and captures the organization's operational needs in the form of epics. This backlog is the primary source of operationally desired capabilities. The release backlog comprises a subset of the program backlog, and includes the features that have been allocated to a specific release. A Sprint backlog is derived from the release backlog, detailing technical specifications (user stories), and may be specific to a single development instance, or Sprint. Synchronization of the backlogs is fundamental to the requirements

Practices for Evolving Requirements and Reflecting Regularly

- **Manage requirements with a program, release, and Sprint-level backlog**
- **Balance technical and programmatic needs with business value**
- **Define a clear owner of the backlog (at all levels) who will be responsible for approving changes**
- **Capture changes identified by the user community as a new user story and include them in the backlog**

management process in order to ensure proper configuration control and understanding of the operational needs, creating a trace between epics in the program backlog to a specific user story in the Development Team's backlog.

Face-to-face engagement with the user/customer is absolutely necessary, as it reduces the risk of incorrectly capturing or developing requirements [24]. Continuous engagement ensures that the prioritization of certain capabilities (user stories) is elicited and captured in the backlog, considering the customer's needs and cost and risk estimations. Traditional requirements management treats functional and non-functional requirements as equals, where the definition of done is implementation of all requirements. In Agile, requirements are prioritized based on the mission value, time criticality, risk reduction/opportunity exploitation, and effort. There are two levels of prioritization: 1) prioritization from a program level and 2) prioritization on a local (team) level, which reflects the goals of the development. The prioritization of user stories is a reflection of the user community/customer's values, including business value and urgency, and is used to manage the backlog. There are several Agile prioritization techniques that may be utilized for this activity, including priority ranges based on business value and urgency and Weighted Shortest Job First, which balances business value with cost of delay [25], [26].

There are clear distinctions of ownership for these backlogs, where the owner is responsible for solicitation and maintenance. The ownership is divided between the organization, including a Line of Business within the FAA, and the developers, including contractors. The FAA must own and develop the program backlog, as it represents where the organization will be going toward in the future. The FAA will be responsible for maintaining that program backlog and working with user communities to identify priorities for the agency. From the program backlog, the program-level manager and the developer will derive their lower level features and user stories, depicting more technical details about the features, as captured in the release and Sprint backlogs. This hierarchy promotes a common understanding of the business needs among all parties. As continuous feedback is an integral part of the Agile process, user stories in the Sprint backlog may be updated or new user stories may be added based on input from the user community and/or Product Owner. All parties must actively manage changes to a backlog to ensure synchronization from a features and priority perspective. Figure 3 5 details the relationship of these various backlogs and the continuous verification of the needs through continuous user engagement.

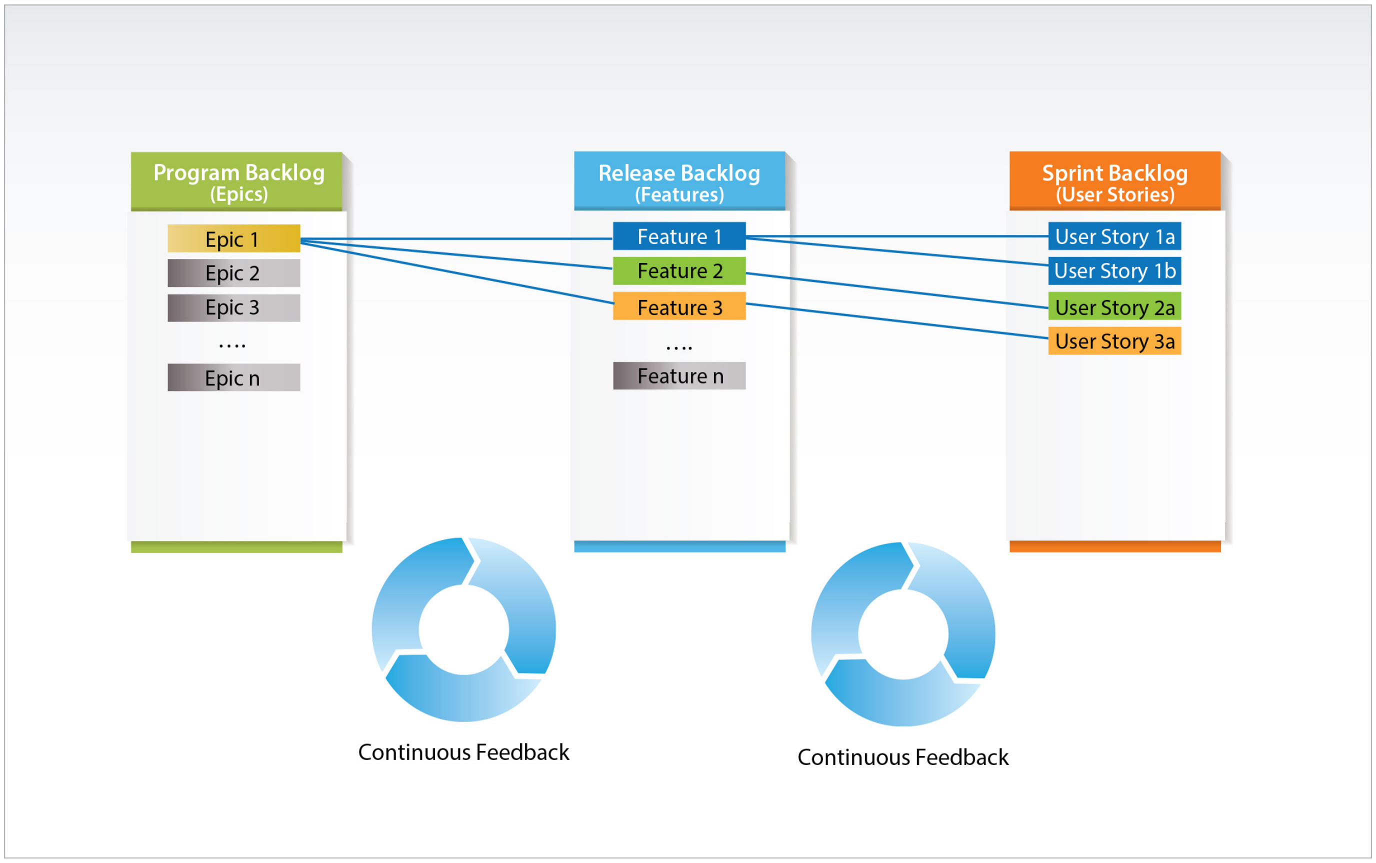


Figure 3-5 Requirements Backlog Management

Requirements management in an Agile context is iterative in nature, with an ongoing refinement of epics, features, and user stories and their prioritization at all levels. Software tools, including JIRA, may help with the management of user stories, ensuring traceability and highlighting changes in requirements. In addition, a requirements traceability matrix could be used to track the source of the epic and user stories and as a tool to assist in the continuous testing and release process [27], [28].

3.3.3 Agile Requirements Challenges

Agile Acquisition enables incremental delivery of value to the user community, and Agile requirements are part of the Agile acquisition process. Agile requirements focus on adapting to the dynamic environment and evolving requirements in a more time-constrained manner, continuously soliciting requirements and validating the requirements. The new Agile requirements engineering process is fundamentally different from traditional acquisition methods. Requirements are initially solicited during the Concept and Requirements Definition phase, refined through Investment Analysis, where they are finalized at the Final Investment Decision, and then validated in Solution Implementation, where the developer will derive program-level requirements into subsystem requirements.

The following challenges must be addressed to adopt an Agile requirements analysis process:

- **Having an owner or team of owners for the vision/epics:** Currently there is some inconsistency in ownership over a particular business need for NAS and non-NAS programs. There must be a dedicated business owner or team who will define the agency's operational environment in a specific time period and the operational capabilities that may support the vision. The owner is expected to guide the direction of development and have the power to make certain requirement determinations that are in alignment with the user community. A potential approach to address this challenge is to identify FAA business owner(s), empowering those who are involved in creating, communicating, and managing strategic vision plans, such as a portfolio manager for NAS capabilities.
- **Defining lightweight user stories:** Requirements written today are often over-specified, with the focus on the technical aspect ("how") instead of the business needs ("what"), potentially encroaching on technical design details during initial phases of the acquisition process. User stories should provide just enough detail for the Development Team to act upon for Sprint planning, while promoting design flexibility. Regardless of the approach, defining requirements at the appropriate level of technical detail requires discipline. User stories provide a framework to help characterize the technical specifications in plain language and describe why the feature is being implemented (what value is being created) rather than how the feature will be implemented. Communicating the business values to the technical developers using a common language that captures the statement of intent (user stories) is one way to ensure the creation of lightweight user stories.
- **Managing changes in requirements:** Current programs experience requirements volatility (requirements creep) due to misalignment with users' needs. Given the changes in the operational environment and long development cycles, the requirements analysis process must accommodate and manage changes in priorities and needs. The GAO reported the following challenge with adapting and applying Agile: "Teams had difficulty managing iterative requirements" [12]. While the backlog facilitates the management of requirements, continuous backlog grooming is a necessary part of the requirements management process. The continuous backlog refinement is a step of the change management process. A strong change management process, which may include establishing a change review board to review and approve enterprise changes, must be established to ensure backlog changes are in alignment with the agency's visions.
- **Balancing urgent fixes (technical needs and concerns) and business values in the backlog:** Release and Sprint planning must account for these competing priorities and ensure that the team can deliver features at the end of a release. The challenge lies in ensuring that certain functional gaps that were identified during the end of a Sprint (during demonstration) are addressed, along with continuing progress toward meeting the release objective(s). As part of the change management process, these functional gaps must be prioritized and placed in the backlog. There must be an agreement among the Agile Team during release and Sprint planning to ensure that the high-priority items, including the urgent fixes, are effectively addressed.

3.4 Agile Cost and Effort Estimation

Programs use cost estimates to enable management to make sound decisions on program alternatives, develop program budgets, and determine program lifecycle costs.

The budgeting process within the government requires for capital expenditures an estimate of the total cost of the program, which must be completed up front. The validity of an Agile estimate for use in the investment analysis process has been a source of controversy, and as result a hybrid approach is appropriate, since the purpose of an estimate changes over time. There are three distinct areas where an estimate of the costs associated with performing work is needed:

1. Budgeting – defines the cost associated with defined activities. In the initial stage of a program, the costs are dependent upon the estimated length of an activity and the resources assigned to complete these actions. This estimate would be expressed as a Rough Order of Magnitude (ROM). Initial budgets are sometimes based on the availability of specific amount of funds and tend to ignore the cone of uncertainty.
2. Planning – defines an approximation of effort and duration based on the size and nature of the effort. The estimate is focused around the cone uncertainty. For Agile, this at the release level.
3. Execution – defines tasks and allocates resources at the Sprint level. Focus is on a much smaller range of tasks and uncertainty.

An estimate by definition is always uncertain and therefore should always be expressed as ranges of uncertainty. Simply stated, the "cone of uncertainty" is the amount of risk associated with an estimate. The level of uncertainty decreases over time as the project definition increases due to a better understanding of the work, and the team performing the work documents their actual velocity. Figure 3 6 shows this relationship.

It is important to understand the definition of cost estimating from the GAO publication, *GAO Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs* [29] when considering estimation in an agile environment. It states,

> *"Cost estimating involves collecting and analyzing historical data and applying quantitative models, techniques, tools, and databases to predict a program's future cost. More simply, cost estimating combines science and art to predict the future cost of something based on known historical data that are adjusted to reflect new materials, technology, software languages, and development teams."*

This estimation has traditionally been done within the FAA as part of the investment analysis effort. A series of documents, starting with the development of a ROM estimate in the Concept and Requirement Definition phase of AMS, refines the estimate over time until at the Final Investment Analysis Review, the program's cost, schedule, and performance are included in the program baseline. This further refinement is necessary since a ROM is only a conceptual estimate done when the scope of the effort is not fully defined, and resource availability and risks are not fully understood. It is useful in choosing between alternatives at a high level for fiscal budget alignment. A credible cost estimate is required to develop a program budget and allow for trade-offs between alternatives.

However, implementing Agile for government projects presents several challenges. GAO's estimation

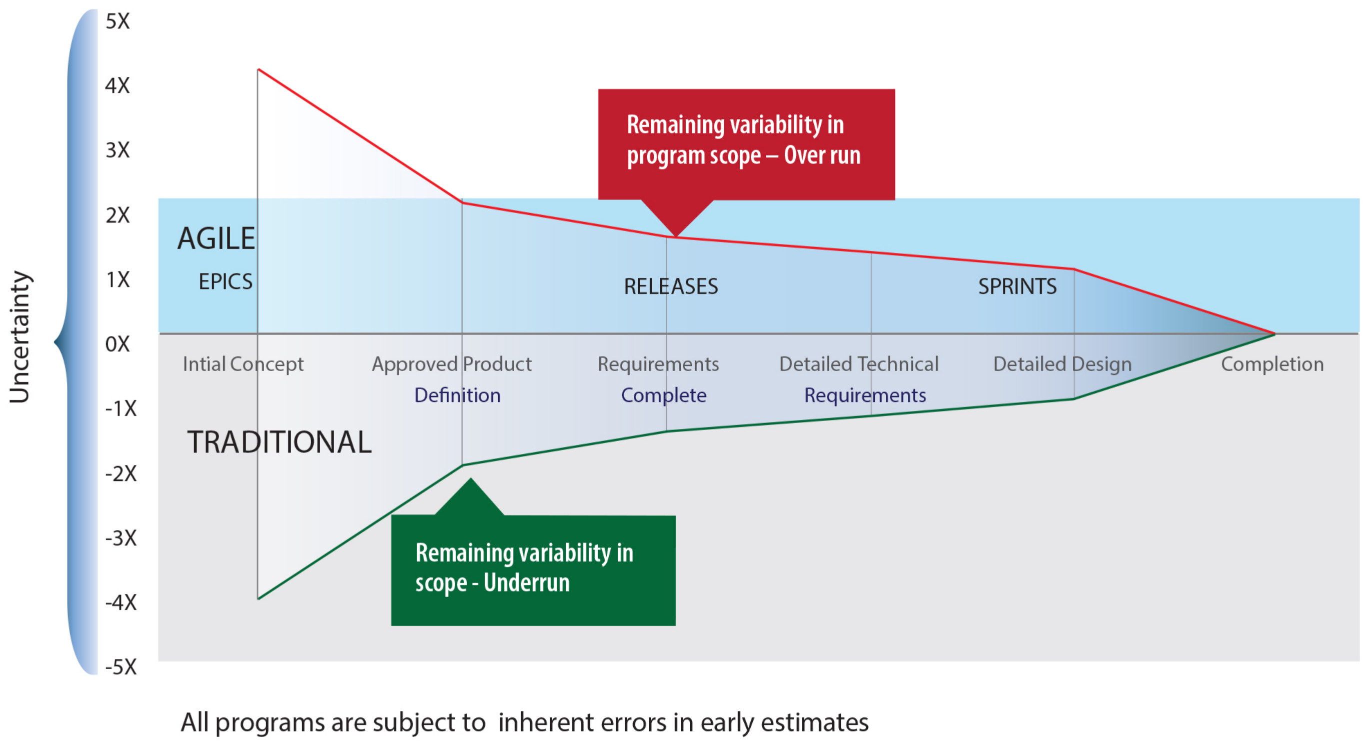

Uncertainty
5X
4X
3X
2X
1X
0X
-1X
-2X
-3X
-4X
-5X
Remaining variability in program scope – Over run
AGILE
EPICS
RELEASES
SPRINTS
Intial Concept
Approved Product Definition
Requirements Complete
Detailed Technical Requirements
Detailed Design
Completion
TRADITIONAL
Remaining variability in scope - Underrun
All programs are subject to inherent errors in early estimates

Figure 3-6 Cone of Uncertainty

process [29] specifies certain steps, including sensitivity analysis and risk and uncertainty analysis, in order to provide a rational estimate and a detailed audit trail. In an Agile context, some of these steps may not be completed due to lack of a detailed baseline solution prior to the commencement of work. In addition, providing a risk-adjusted cost estimation, as currently required at the Initial and Final Investment Analysis Decision points, may not be practical due to the relative measures used in Agile estimation. As a result, there is a need for a revised estimation process that is more suitable for Agile.

The key Agile Estimation principles are as follows:

- Evaluating cost based on complete work performance
- Increasing fidelity of work estimations over time.

3.4.1 Evaluating Cost Based on Complete Work Performance

Practices for Evaluating Cost Based on Complete Work Performance

- **Estimate program costs based on traditional cost estimation processes, influenced by established or prospective pace of Agile development**
- **Refine estimates after each program increment, based on development of performance retrospectives**

Cost estimation techniques do not differ greatly for Agile development programs and those created for traditional development programs. Regardless of the development type, the cost estimation must still consider development costs and the costs associated with managing, acquiring, and maintaining a program. The main distinguishing factor is the use of relative value of efforts for estimating the cost of Agile development, where the Development Team effort estimations will inform the program cost estimates upon completion of a Sprint and/or Release.

At first glance, the tenets of an Agile development may seem in conflict with today's government budgeting process, where IT investments are currently appropriated based on a detailed business case and are typically of longer duration, 18 months or more, with six-month releases and many simultaneous Sprints per Release. In addition, government investments have extensive enterprise architecture, security, system integration, and oversight requirements, which may impede the ability to support rapid development. However, Agile may actually be better suited to the government budget process, since the program can deliver on a "build to budget" concept as fiscal budgets are fixed several years ahead of the actual scheduling and analysis of the work. For example, the program manager may plan the program around the concept of "affordability," taking into consideration the fixed development budget and the prioritization of the work for each period of performance (time-boxed development effort). This prioritization would be based on the implementation of the Agile planning concepts discussed in Section 3.2., Agile Planning. The Agile planning team must work closely with all stakeholders to align expectations, particularly concerning budgeting and oversight responsibilities.

Mike Cohn [30] laid out the basics of the Agile cost estimation, which are the basis for most Agile estimations. The following are his three concepts:

1. *"Estimation of overall size can only give a high-level estimate for the work item, typically measured using a neutral unit such as story points.*
2. *Velocity is a measure of how many points this Agile Team can deliver within an iteration.*
3. *Estimation of effort for a work item translates the size (measured in points) to a detailed estimate using hours for each subtask. This estimation is usually undertaken at the beginning of a sprint or iteration, and is based on the velocity and whatever heuristics have been used to characterize the user stories that are the basis for the work item."*

Agile planning and estimating is initially performed at the high level, with detailed planning done at the lowest level. As shown in Table 3 1, adapted from the National Defense Industrial Association [31], the precision of the estimate increases as the time performance timeframe decreases.

Table 3-1 Cost Estimation Precision

PRECISION	LEVEL	FREQUENCY	HORIZON	LEVEL	ARTIFACT
Lowest	Program	Program Startup Updates throughout	Program Duration	Epics Features	Program Backlog Product Roadmap (including the definition of the Minimal Viable Product)
	Release Planning	Each Release	Release	Feature User Stories	Program Backlog Updates Release Backlog
	Sprint Planning	Each Sprint	Weeks	User Stories Tasks	Release Backlog Updates Sprint Backlog
Highest	Daily Planning	Daily	Day	Tasks	Update Sprint Backlog

Agile cost estimation is product based and completed in a series of iterative activities where the level of detail increases as the program proceeds. The initial performance baseline for a program is established at the highest level, consisting of these artifacts: product vision, product roadmap, initial product backlog, and the minimum viable product, as described in Section 1.3.3. This initial estimate would be suitable to use for budget planning purposes.

Agile cost estimations at the release planning level should be to a level of detail that supports the analysis of the alternatives contained within the business case and the development of the documentation needed for the FAA's annual budgeting process. This high-level estimation may be based on the features that have been allocated to a particular release, evaluating the level of work to be performed using traditional parametric models that are described in the following section. Using these products and an estimate of the productivity rate (velocity) of the team based on historical data and the size of the team, the cost of the product can be estimated and allocated into a time-phased budget for planning and completion of the Investment Analysis.

Finally, as the program enters the Solution Implementation phase, detailed planning occurs at the Release and Sprint levels. Since these releases will be planned closer to the actual execution point, the quality of the estimate and the program's ability to perform within the parameters will improve. The cost estimation at the Development Team level, or Sprint level, may be done using the following three equations:

1. Identify the total number of user stories (US) and story points (SP) per user story and compute total story points (TSP): TSP=US*SP
2. Compute the estimated development time (EDT) for the program by dividing TSP by Velocity (V): EDT =TSP/V
3. Multiply EDT by the average cost of the Agile Development Team to determine total development cost (TDC): TDC = EDT*$.

The Program Office will re-compute the cost estimate for each subsequent budget year by subtracting the delivered user stories from the product backlog to create a revised TSP and update the Velocity value based on actual team performance in the current year to determine the development costs remaining. Planning activities at the Release and Sprint levels provides an opportunity to update the development cost effort, thus continually updating and refining the development cost estimate as time progresses.

Agile cost estimation at the program level requires aggregation of the costs associated with each of the Sprints. One approach is to use the methods discussed in the Scaled Agile Framework, which uses the concept of a portfolio vison consisting of business and architectural epics and Agile Release Train, which is team of Agile teams that delivers program-level value [32], where the development effort takes place. These epics and releases are managed through the backlog process.

Cost estimation is a combination of engineering and economic activities leading to a program cost estimate. While there are many approaches to develop a cost estimate, the flow chart in Figure 3 7 represents the steps generally used.

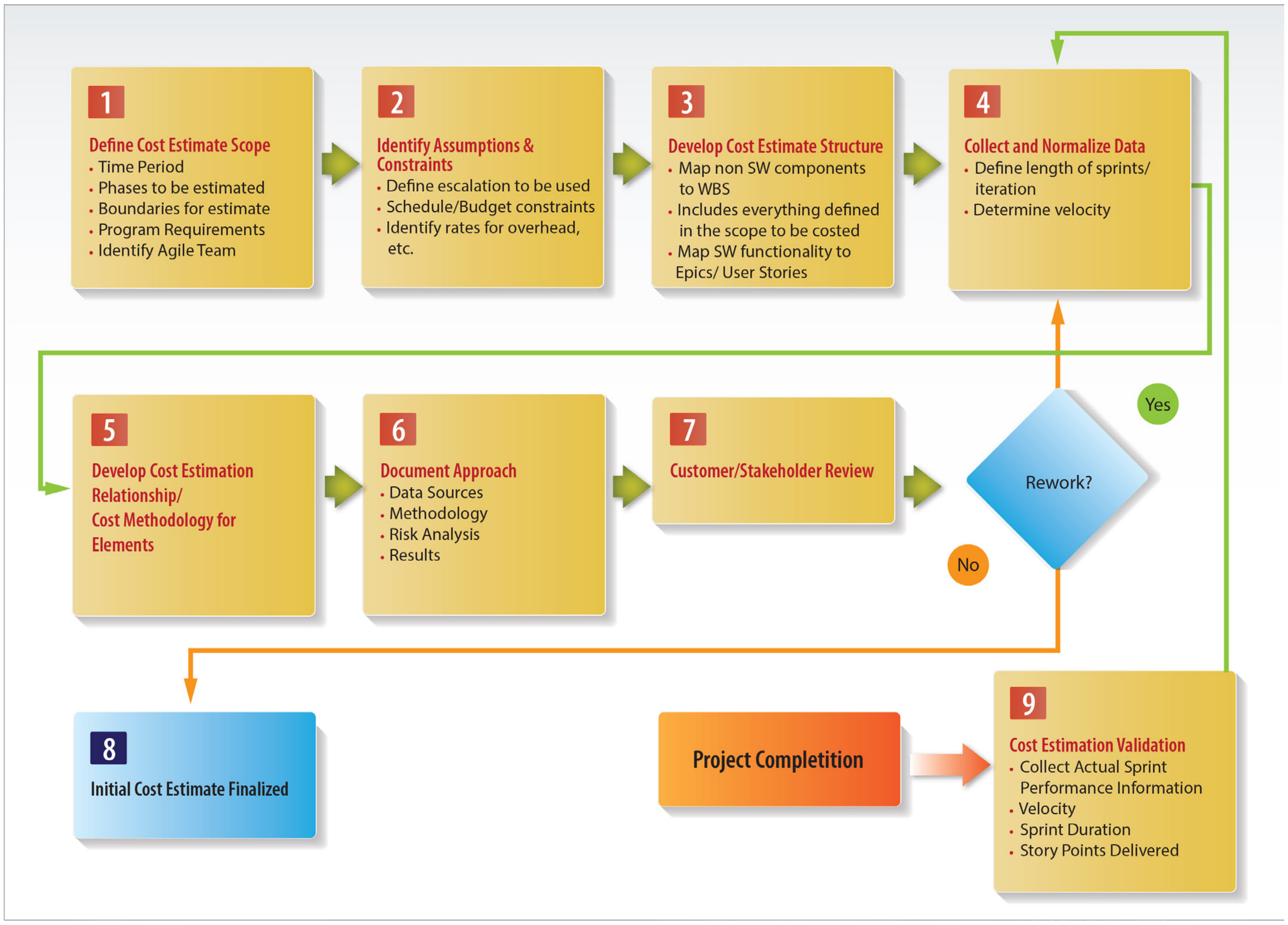

1
Define Cost Estimate Scope
• Time Period
• Phases to be estimated
• Boundaries for estimate
• Program Requirements
• Identify Agile Team
2
Identify Assumptions & Constraints
• Define escalation to be used
• Schedule/Budget constraints
• Identify rates for overhead, etc.
3
Develop Cost Estimate Structure
• Map non SW components to WBS
• Includes everything defined in the scope to be costed
• Map SW functionality to Epics/ User Stories
4
Collect and Normalize Data
• Define length of sprints/ iteration
• Determine velocity
5
Develop Cost Estimation Relationship/ Cost Methodology for Elements
6
Document Approach
• Data Sources
• Methodology
• Risk Analysis
• Results
7
Customer/Stakeholder Review
Rework?
Yes
No
8
Initial Cost Estimate Finalized
Project Completition
9
Cost Estimation Validation
• Collect Actual Sprint Performance Information
• Velocity
• Sprint Duration
• Story Points Delivered

Figure 3-7 General Depiction of Agile Cost Estimation Process

3.4.2 Increasing Fidelity of Work Estimates over Time

Practices for Increasing Fidelity of Work Estimates over Time

- **Teams should examine completed work against original estimates to validate estimates**
- **Uncertainty is minimized when the level of effort is estimated close to when the work is performed and by the team performing the work**

Cost estimation is dependent on an understanding of the work to be performed. Many of the traditional software estimation tools, such as Constructive Cost Model, Price-S, Software Lifecycle Management-Estimate, and Software Evaluation and Estimation of Resources, have been adapted for Agile software development. The most commonly used methods for Agile estimations are not based on these parametric models, but rather on comparisons with previously completed work and the judgment of the Agile Team performing the work in the past. The following describes these estimation techniques in increasing degree of accuracy:

- Relative comparison – evaluating whether a particular user story is more complex/larger than another (e.g., A is bigger than B and B is bigger than C)
- Historical – evaluating the user story's complexity based on previous similar efforts
- Story points – evaluating the complexity of a user story expressed in points (Fibonacci Series: 1, 3, 5, 8...) or size (small, medium, large...).

Story points have been discussed by the Society of Cost Estimation and Analysis and the Association for the Advancement of Cost Engineers as a method of cost estimation in Agile development [33].

An additional attribute of Agile is that since development efforts are pursued in small increments, the effort is estimated close to when the work is performed and by the team performing the work, so there will be very little uncertainty in the estimation. The level of abstraction for calculating work estimations may be difficult for new Agile teams to understand, as relative user story complexity is not directly correlated to the level of effort needed for development. Refinement will come as the Development Team becomes more adept as the Sprints progress. In addition, retrospectives at the end of a Sprint may help to identify and address estimation issues, including accuracy.

To validate these estimates and progress, the team should use a burn-down chart, which is based on story points, to track the progress of the Agile development effort. These story points may be converted into dollars for cost estimation. As the cost estimates for the Agile portion of the program are based on factors that are themselves estimates, it is very important that the program review the actual costs after completion of the project and use the results to validate and revise the estimating factors for future use.

3.4.3 Agile Cost and Effort Estimation Challenges

There are a few challenges in estimating an Agile program, as follows:

- There is no generally recognized standard unit of measurement for any of the common approaches to Agile estimation. Examples of Agile work estimation have used story points, user stories, and epics. However, all of these items are subjective and dependent upon the experience and skills of the team developing the estimate. Story points, for example, are a relative measure of size against similar work performed by the estimating team in the past.
- Cost estimating is dependent on the composition and expertise of the team. To improve the quality of the estimate, the team that will perform the work should be doing the estimation. Unfortunately, this is not always possible, particularly since no development will have been completed upon the creation of the initial program estimate. Traditional cost estimation techniques may be used to help define the program's initial cost. Care must be taken to ensure that the initially defined cost estimation is in alignment with updated estimations at the Release and Sprint levels.

3.5 Agile Contracting

Agile contracting involves two perspectives: structuring acquisition contracts in a way that effectively enables Agile development, and applying Agile Team and culture principles to make the development and negotiation of contracts more effective. This section focuses on the first perspective. Applying Agile effectively requires considerable planning and coordination between the acquisition organization, the program, and eventually the performing contractor. The first step of that planning and coordination is to consider whether Agile is the most appropriate approach to the acquisition. Agile has been typically applied to programs that require significant software design and development, so it will clearly be an appropriate candidate for such programs. The use of Agile is less appropriate for the procurement of commercially available products, subscription services, commoditized services, or services from other government agencies, for which little or no development activity is expected.

AMS provides the overarching policy and guidance to FAA contracting officers (COs) for the procurement of goods and services, including IT and digital services. The acquisition policy and guidance have been designed to allow the CO flexibility to be innovative and creative to meet the procurement system's goal to "obtain high quality products, services... in a timely cost-effective manner" [34]. AMS does not specify any particular approach to contracts and allows the COs to use their best business judgment when determining what type of contract to use.

Regardless of the development methodology employed, Waterfall model or Agile, it is still necessary for COs to "make acquisition decisions that deliver the best value product or service to the customer" [35]. As COs provide the business judgments, serving as the bridge between the FAA and its industry partners, a close partnership with the Program Office is necessary to conduct an Agile Acquisition, where the CO, who is knowledgeable of Agile, provides the tools for a program to monitor, administer, and terminate a contract.

The Agile principles for contracting are as follows:

- Structuring the contract to support Agile practices
- Defining and leveraging an Agile contracting strategy.

3.5.1 Structuring the Contract to Support Agile Practices

Practices for Structuring the Contract to Support Agile Practices

- **Emphasize an outcome-based approach**
- **Consider the entire acquisition lifecycle for the acquisition strategy**
- **Ensure that the contract supports frequent delivery of capability, visibility into contractor performance, continuous testing, and accountability for results**

Typical FAA contracts [36] are based on a fixed technical scope and involve a significant lead time to prepare and award, often months to over a year. This process, and the fixed contracting requirements associated with it, are potential impediments to an Agile developmental model, and must be re-evaluated to ensure that the contracting approach supports agility throughout the acquisition lifecycle. Contracting within an Agile context requires a fundamental change, moving away from traditional timelines and a need for detailed design specifications for the full solution set. This does not mean that acquisition planning should not be done, but rather that the emphasis changes to a more outcome-based approach, and these outcomes will be expressed at a higher level. It is extremely important that the team consider the entire acquisition lifecycle when developing its acquisition strategy. Agile contracting strategies must enable smaller development and deployment schedules (like Sprints and Releases), while maintaining flexibility and promoting a more streamlined contracting process.

- Agile contracting should support the following acquisition goals:
- Frequent delivery of usable capabilities that provide value to the end user
- Greater visibility into contractor performance
- Risk reduction through frequent testing and user feedback
- Accountability for results through the early identification of problems with adequate time for correction.

3.5.2 Defining and Leveraging an Agile Contracting Strategy

A successful Agile Acquisition requires considerable interaction among the business owner, key stakeholders, the performing organization, oversight, and contracting organizations. To define an appropriate contracting strategy, clear contracting requirements must be differentiated from system requirements. These contracting requirements should not be confused with the traditional development requirements. Contracting requirements specify activities that a contractor must perform. Development requirements specify operational capabilities that address the mission need. For an Agile contracting strategy, the requirements should focus on high-level functionality and associated milestones/schedule instead of specific features. The focus on the expression of higher level capabilities helps to emphasize development outcomes and intent, while providing the flexibility of the Agile process to address potential

Practices for Defining and Leveraging an Agile Contracting Strategy

- **Cost-based contracts (cost-reimbursement or Time and Materials) provide the contractual flexibility that corresponds to the flexibility sought in Agile practices**
- **Firm Fixed Price contracts may be appropriate when development complexity is understood and contractor performance is established**
- **Indefinite-Delivery, Indefinite-Quantity or Blanket Purchase Agreement vehicles provide contractual flexibility that can enable Agile practices**
- **Performance-Based Contracting strategies can enable Agile practices**
- **Ensure that the solicitation clearly addresses Agile considerations**
- **Emphasize expertise in Agile development, and past performance during source selection**
- **Promote constant communication between the CO and program throughout the contract lifecycle**

changing or emergent requirements at lower levels of detail. The contract requirement must reflect the expected outcome (more objective oriented).

The three basic types of government contracts—fixed price, cost reimbursement, and Time and Materials (T&M)—have been designed to allow for the uncertainty in requirements and the risks associated with performance. In a fixed price contract, the price is negotiated prior to award and payment is tied to delivery of a product or service. It is the most appropriate contract type when the government knows in advance what is needed to meet its requirements (e.g., a commercial product or a well-defined software effort). If the government lacks experience with the contractor or uncertainty exists about the complexity of the development effort, a Firm Fixed Price (FFP) contract can be problematic because of the unknowns.

A cost-based contract (either cost reimbursement or T&M) is more appropriate if specific requirements or other project details are incompletely defined at the start of the effort. This could be caused by the uncertainty in the business owner's statement of need or by rapidly changing technology. These types of contracts place greater risk on the government, since the contractor only has to provide its "best effort." Both of these types of contracts place greater burdens on the government to monitor the performance of the contractor to ensure completion of the tasks and to control costs. As the government gains a better understanding of Agile development and effective Agile cost and effort estimation tools are determined, development tasks may become more amenable to fixed price contract approaches. The contract types and association risk levels are shown in Figure 3 8.

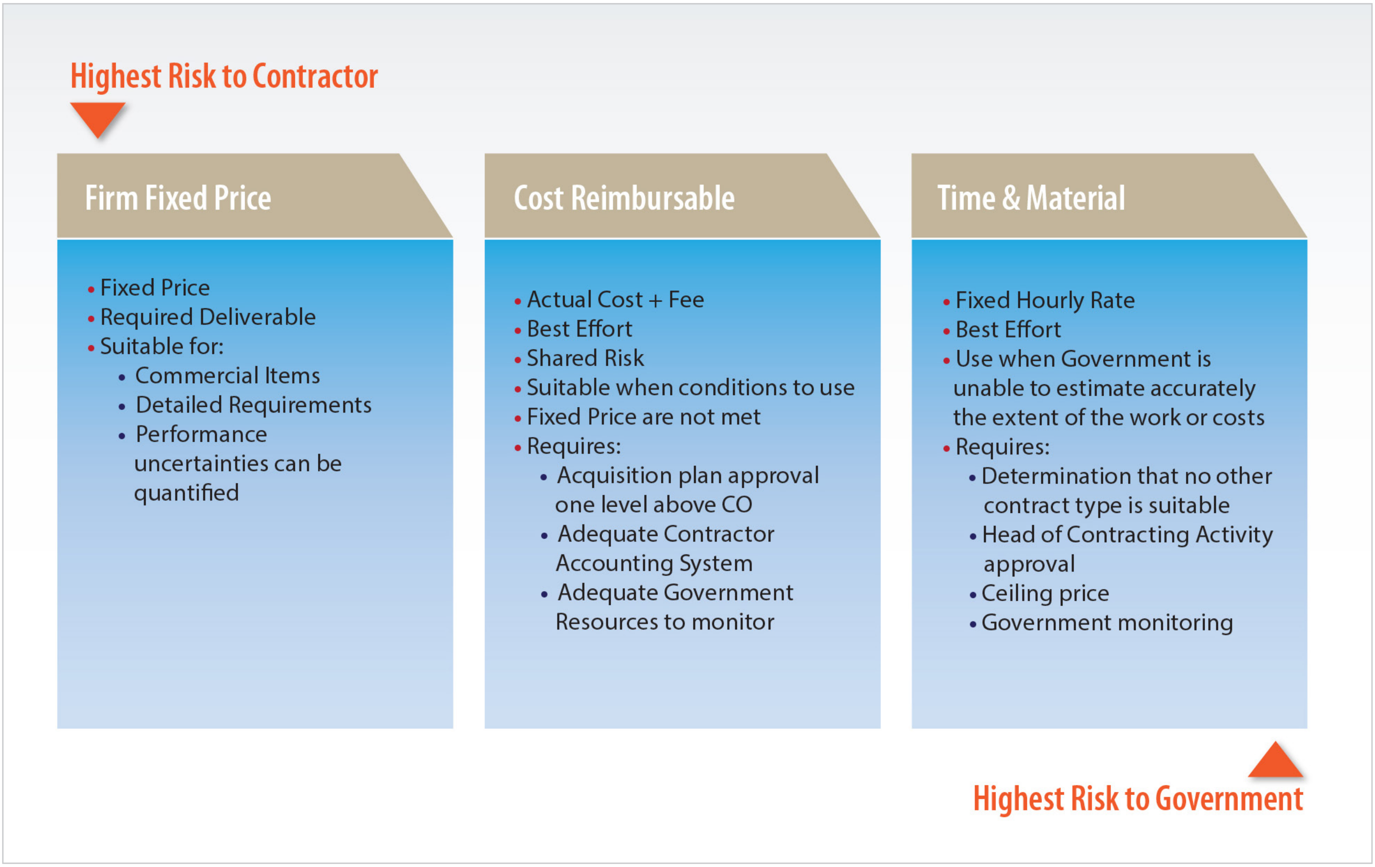

Highest Risk to Contractor
Firm Fixed Price
• Fixed Price
• Required Deliverable
• Suitable for:
• Commercial Items
• Detailed Requirements
• Performance uncertainties can be quantified
Cost Reimbursable
• Actual Cost + Fee
• Best Effort
• Shared Risk
• Suitable when conditions to use
• Fixed Price are not met
• Requires:
• Acquisition plan approval one level above CO
• Adequate Contractor Accounting System
• Adequate Government Resources to monitor
Time & Material
• Fixed Hourly Rate
• Best Effort
• Use when Government is unable to estimate accurately the extent of the work or costs
• Requires:
• Determination that no other contract type is suitable
• Head of Contracting Activity approval
• Ceiling price
• Government monitoring
Highest Risk to Government

Figure 3-8 Contract Types and Associated Risk Levels

A project could also be awarded as a hybrid type of contract, where some items would be structured as FFP and others as cost based. The CO could then use the appropriate payment strategy depending on the level of certainty of the requirements. For example, an FFP contract could be used for the hardware or commercial off-the-shelf software component of a system, while a cost-based contract could be used for customized software development. An FFP component can be highly effective for a Sprint of limited duration when the effort is under the control of a single contractor. A cost component would be more effective if a team effort is being used when multiple contractors produce parts of the solution.

Contract vehicles are another key component of a contract strategy. Some examples include using single or multiple Indefinite-Delivery, Indefinite-Quantity (IDIQ) contracts or a Blanket Purchase Agreement as the primary contract vehicle, with task orders to capture the immediate program development needs (Sprints). The intent behind using these contract mechanisms is to provide flexibility and enable faster execution. For example, a multiple IDIQ award allows multiple Agile developers/contractors to compete at the individual task order level, and should the contractor not deliver the expected objectives by the end of the award period, the government (FAA) is free to select a new Agile contractor for the next iteration. The Office of Management and Budget (OMB) provides guidance on modular contracting [37] through a checklist for aligning contracting practices. The guidance recommends that contracts be flexible to adopt changing needs and an iterative development cycle. Agile is ideally suited for Performance-Based Contracting, since the focus is on delivering functionality rather meeting complex requirements.

A successful solicitation for Agile will require that the contracting staff and program team be part of a highly interactive process throughout the contract lifecycle, where the documentation may be different than what was delivered using the Waterfall approach. It is essential that the team develop an Acquisition Plan that clearly addresses these items:

1. Statement of Need
2. Product or Service Descriptions
3. Estimated Cost
4. Risks
5. Source Selection Considerations
6. Non-functional Requirements
7. Quality Assurance Methodology.

The TechFAR [10] recommends that as part of the solicitation, the CO should explain that the contract will be based on Agile development methods and the level of FAA interaction with the contractor as part of the Agile team. The solicitation should also clearly address Agile considerations as part of the contractor's proposal, including the development of user stories, integration testing, non-functional requirements, and acceptance criteria.

A best value approach to source selection is the most advantageous method of contractor selection, since success in Agile development is dependent on the skills and expertise of the contractor staff. The development of evaluation criteria is the key to any successful solicitation and is even more important when using an Agile development method, since the FAA will be procuring the services of a company based on the skills of its development team and company experience. The technical evaluation factors should focus on the following:

1. Contractor interaction with the FAA during development activities
2. Management approach, including performance metrics
3. Alignment of product roadmaps or release strategy to the product vision
4. Contractor's experience with Agile and staffing approach
5. Quality control process.

In acquiring contractor services for an Agile development, the acquisition team should focus on conducting the appropriate market research to determine which companies possess the necessary Agile development skills, have experience in delivering software to government agencies, and have the necessary business processes to perform cost reimbursable work. Additionally, past performance can be a key factor in selecting a contractor, since many Agile development efforts in industry are not at the level of complexity, safety criticality, size, or scope of many FAA programs, particularly those related to the NAS.

In all, Agile contracting requires discipline to ensure that the appropriate program objectives are met. Utilizing Agile-oriented common contracting templates and contracting requirements may be beneficial when defining a contracting strategy for an Agile Acquisition. However, continuous communication between the CO and the program manager is the underlying foundation for success.

3.5.3 Agile Contracting Challenges

Agile contracting is an essential component of enabling Agile Acquisition. Specific challenges to Agile contracting include the following:

- Developing a common understanding of Agile techniques by the development and acquisition teams so that an acquisition strategy can be properly structured. Common understanding depends on effective training and collaboration between developers and the acquisition team.
- Assigning contracting personnel who are capable of assisting the program in making the business decisions and trade-offs that come with the implementation of an Agile effort. This challenge may be addressed by ensuring that contracting personnel are adequately trained in Agile processes and dedicated to specific acquisition(s).
- Ensuring that the program makes a commitment to share information freely across internal organizations and with the developing contractor. The challenge might be addressed by having a communication plan that defines from the beginning the processes and mechanisms to be used to ensure coordination.
- Committing to the Agile effort in the beginning and remaining committed for the duration of the program lifecycle, since the Agile effort is based on small teams whose membership remains constant over time. The conduct of pilot programs can demonstrate a commitment to Agile efforts.
- Adopting an efficient approach to the acquisition effort. If a large time burden is imposed on the acquisition team for reviews and constant refinement of documents, then its ability to implement an Agile program will be hindered. The challenge might be addressed by refining a potential Agile lifecycle, such as the one presented in Section 2, that establishes expectations regarding needed documentation, review processes, and decision making for Agile efforts.
- Embracing the organizational change required to foster a culture based on the concept of frequent delivery. Pilot programs can be an effective way to adapt Agile to the organization and demonstrate the value.

3.6 Agile Test and Evaluation

T&E is the process by which a system or components are compared against requirements and specifications through testing. The results are evaluated to assess the progress of design, performance, supportability, etc. Developmental test and evaluation is an engineering tool used to reduce risk throughout the acquisition cycle. Operational test and evaluation (OT&E) is the actual or simulated employment by typical users of a system under realistic operational conditions. [38]

T&E is an integral part of Solution Implementation and involves evaluating a product from the component level, to stand-alone system, integrated system, and if appropriate, system-of-system and enterprise levels. Testing is traditionally conceptualized as following development in a sequential process, as depicted at its most basic in Figure 3 9. The reality of software development is that testing occurs at the software unit or component level, the lowest level of testing, as the code is developed. Unit tests are usually written by developers as they work on code to ensure that the specific function is working as expected. Depending on the organization's expectations for software development, unit testing

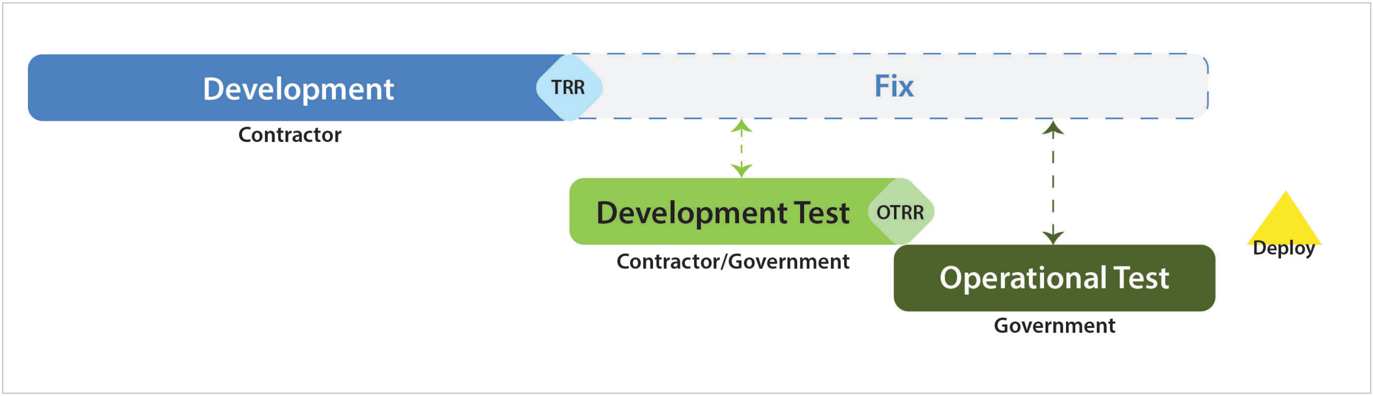


Figure 3-9 Traditional Conceptualization of Testing in Acquisition

might include static code analysis, data flow analysis, metrics analysis, code coverage analysis, or other software verification practices. Unit integration and associated integration tests may occur episodically, with some organizations conducting integration on a daily basis (e.g., nightly builds). Still, the more traditional approach to testing involves a testing and integration phase after completion of development. For government programs, testing may be segmented into the aforementioned contractor-oriented development testing and government operational testing. Each phase of the process culminates in a readiness review testing milestone (Test Readiness Review [TRR] or Operational Test Readiness Review [OTRR]) to ensure proper preparation to enter the next phase and minimize the likelihood of defects necessitating a return to the prior phase.

The T&E strategy is formulated by the government in the planning stages of the program, which precedes the Solution Implementation phase. The strategy will be reflected in products such as the Implementation Strategy Planning Document (ISPD) and the TEMP, which documents the overall structure and objectives of the T&E program. The Statement of Work (SOW) will impose requirements on the contractor to plan and conduct a test program, including documentation such as the contractor TEMP and test plans, procedures, and reports.

The FAA provides substantial guidance for the scope and conduct of T&E activities, including the following:

- AMS policy Section 4.4, Test and Evaluation [39], defines the role of T&E in each phase of the AMS lifecycle.
- AMS policy Section 4.5, Independent Operational Assessment [40], identifies the responsibilities of FAA authorities to determine the need for independent operational assessment and to provide determination of system operational readiness in support of production and in-service decisions.
- The Test and Evaluation Process Guidelines [41] provides a foundation for planning and executing T&E activities appropriate to each individual investment program.
- The Test and Evaluation Handbook [42] provides detailed guidance.

An Agile approach to T&E involves re-conceptualizing how and when testing is applied in the development process. Important Agile T&E principles include the following:

- Satisfy users through collaboration.
- Test delivered software.
- Maintain a constant pace of testing.
- Evolve tests with the software design.
- Measure working software.

Many additional resources are available to advance understanding and application of Agile principles in T&E [41], [42].

3.6.1 Satisfying Users through Collaboration

Practices for Satisfying Users through Collaboration

- **Time-box activities within the planning phase**
- **Ensure that requirements encompass test objectives**
- **Align test reviews with Agile development cadence**
- **Consider test-driven development**
- **Document test results as necessary**

Agile principles influence the Agile approach to T&E in several ways. First is the importance of up front engagement and collaboration of the test community with the stakeholders and Development Team during strategic planning. Ideally, testers will be integrated with the Development Team in the earliest stages of planning and continue to be integral throughout development and implementation. For larger efforts, it may be necessary to have distinct test teams working in parallel with the product Development Team. Test personnel need to be involved early in the planning stages to ensure that testing perspectives are captured in the estimation of required resources, in scheduling, and in risk identification and mitigation. Testing perspectives will be captured in planning documents such as the ISPD and the TEMP, and the developer's expectations will be reflected in the SOW. At a tactical level, testers work with users to ensure that user stories have a definition of "done" and acceptance criteria that are testable. Testers may define test user stories to encompass the work needed to develop tests and procedures and work with the Product Owner to ensure that priorities reflect the dependencies of testing on software completion. Testers are actively involved with users in ensuring the operational suitability of a delivered release prior to deployment. Other Agile influences on planning phases include the following:

- Time-box activities within the test planning phases. The traditional FAA execution of the analysis phases can take multiple years. Adopting the Agile principle of establishing a fixed duration for an activity (e.g., Sprints) can promote focus, particularly with dedicated teams, reducing the effort expended on lower priority outputs.
- Ensure that the program requirements or corresponding Agile epics encompass test objectives.
- Accept that planning products such as the ISPD and TEMP are dynamic products that need to

evolve over time. Agile documentation principles suggest that the information involved be readily accessible to all team members and stakeholders and that the content be tailored and prioritized to provide information of most value to the program.

- Plan design and test reviews to be periodic rather than one-time events. The initial design review will be high level, referring to the epic(s) associated with the current development activity. More detailed reviews should be conducted in planning for each Release, updated as necessary at the start of each Sprint. Test plans are an integral part of the Release planning activity, and content should be tailored to meet the "just enough" attribute of Agile documentation practice.
- Define test procedures in conjunction with user stories allocated for development in each Sprint. The test procedure is necessary to define the success criteria for acceptance of the function/feature involved. At the extreme, the test procedure might precede and drive the design (i.e., Test Driven Development [TDD]).
- Tailor reporting of test results to the needed information. During Sprints, representative of test outcomes are the completion and acceptance of functions/features and the capture of defects to be fixed or enhancements to be pursued in the backlog. Test planning and reporting for final release integration and operational evaluation warrant a more rigorous approach that needs to be captured in release plans.

3.6.2 Testing Delivered Software

Practices for Testing Delivered Software

- **Promote common infrastructure**
- **Automate test procedures**
- **Include test procedure and function development in backlogs**
- **Use a common trouble reporting/tracking mechanism**

In the Agile model, testing occurs in line with development during Sprints, which define the pace of software development and testing. Several levels of testing are involved. Software developers typically conduct unit tests for the software items being developed, similar to the traditional developmental approach. The unit tests ensure that the function/feature being developed meets the operational intent. The definition of "done" includes passing the associated test, enabling credit to be claimed for the software function embodied in the unit. These unit tests may occur at any point in the Sprint as the software unit developments progress. A best practice is to test the code each time it is checked in (i.e., continuous test and integration). Agile discipline often requires that completed units for which credit is to be claimed will be integrated into the software baseline at or before completion of each Sprint. This will involve a higher order of testing to ensure that new or modified functions/features integrate effectively with existing system functionality. The Agile process culminates in a system demonstration, which provides opportunities for user acceptance and feedback regarding system effectiveness and suitability.

The early user exposure and feedback differentiates Agile from traditional T&E—it effectively combines elements of traditional T&E and OT&E, sometimes characterized as combined test and evaluation.

The Agile approach to T&E may share with the traditional approach the need for a specific T&E phase at the completion of Release development. One or more Sprints may be specifically dedicated to this kind of testing. This testing will involve the integration of all the new features with features that may have preceded the current Sprint. Regression testing may be conducted to ensure that performance continues to be satisfactory. Special considerations such as cyber security or software design assurance for safety certification may also influence the level of testing involved. Figure 3 10 illustrates this incorporation of T&E within an Agile development lifecycle. Alternatively, a mature Agile process may integrate automated testing with an automated build process such that integration and regression testing is done continuously, minimizing the need for the Release T&E phase.

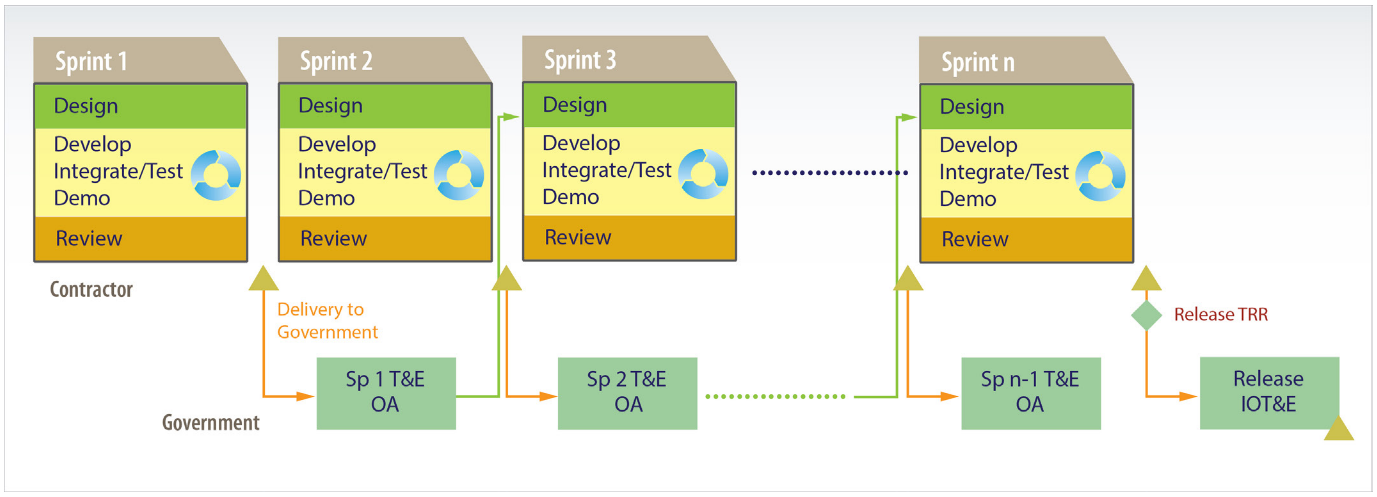


Figure 3-10 Agile T&E Integration

To promote the effectiveness of Agile development, a number of additional practices should be employed. The T&E strategy should include investment in T&E assets that enable parallel software development and testing. Automated test procedures are important to ensure that necessary tests can be accommodated within Sprints, and development of test procedures needs to be included in the program backlog. Ideally, the government should establish the test infrastructure at a portfolio or enterprise level, leveraging the test infrastructure use across multiple programs and ensuring that component programs are tested more effectively and efficiently. There should be common shared bug/defect tracking process among developers and testers to promote a common understanding of the defects, their priorities, and the integration of their resolution in backlog management.

3.6.3 Maintaining a Constant Pace of Testing

Practices for Maintaining a Constant Test Pace

- **Align testing with Agile development cadence**
- **Consider test-driven development**

Effective Agile execution requires that testing be integrated with software development. The allocation of Sprint development tasks must include the associated testing to verify software completion (i.e., function operation and suitability), so the development of test features must correspond to function development and influences the overall development pace. In the approach of TDD, development of the test approach and acceptance criteria should precede and inform the development of the function/feature involved. To complete the alignment with Sprint pacing, integration and operational testing of the complete release may be conducted as one or more test Sprints dedicated to the purpose. Ideally, the test Sprints should occur in parallel with the development (i.e., test the previous Sprint during the subsequent Sprint) rather than waiting until the very end. The conduct of testing should be as close to the development cycle as practicable to identify and rectify issues early.

3.6.4 Evolving Tests with the Software Design

Practices for Evolving Tests with Software Design

- **Adapt tests to changing requirements and software design**
- **Automate test procedures**

Agile eschews comprehensive definition of requirements up front in favor of flexibility and adapting to change. Adapting to changing requirements, functional and non-functional, demands associated software design changes. Test capabilities must also change to accommodate the changing design. Consideration must be given to test software configurations to ensure that testing remains current and relevant to evolving releases.

3.6.5 Measuring Working Software

Practices for Measuring Working Software

- **Plan for and conduct post-implementation reviews**

Verifying function and performance of delivered software is a traditional focus of testing, but verifying benefits and validating objectives of the application development are often indefinitely deferred or overlooked. Testing should include capabilities to measure function use and system performance in operation that enable determination of whether and to what degree expected operational benefits are being realized (e.g., NAS post-implementation reviews). Measurements may be used to suggest how operations might be tailored or training approaches enhanced to better use provided capabilities.

3.6.6 Agile Test and Evaluation Challenges

Testing is challenging in any development. Specific challenges to Agile testing include the following:

- Establishing and maintaining concurrency of the operational baseline and test configuration, as described in conjunction with the principles of Test Delivered Software and Evolve Tests with the Software Design. This might involve adopting "continuous" integration practices such as daily builds that promote incremental enhancement of the baseline while ensuring stability.
- Potential complexities of integration testing and system of systems. Integrating functions during Sprints may involve only project-specific assets, with external interfaces driven by simulations. Exposure to operational interfaces may not occur until testing a Release in preparation for deployment or, in the extreme case, until the Release becomes initially operational. One way to address the challenge might be to expand the test assets and capabilities of FAA test organizations to permit greater access to "operational" interfaces and to a higher fidelity test environment.
- Realizing common enterprise infrastructures. Development and test of software for the enterprise can leverage economies of scale and reuse by promoting common infrastructure and standards that span multiple projects. The objective could be advanced by adopting more prescriptive technical standards (e.g., specify an FAA common operating system such as Linux); this could be particularly effective when a new IT infrastructure is introduced, such as cloud-based applications.

3.7 Agile Deployment and Sustainment

Deployment involves the introduction of systems, services, or capabilities into operation. The operational context may vary in any of several ways, including initial installation or implementation, introducing new capabilities into the existing infrastructure, maintaining the existing capabilities and infrastructure with bug fixes and updates of system elements, or larger scale updates of capabilities or infrastructure such as technology refresh and/or refactoring of software designs to address failing elements or technology changes. Although the focus of Agile deployment and sustainment is the development and management of the software that drives system capabilities, considerations necessarily involve hardware deployment and system integration, training of users and system operators, logistics support, and other factors fundamental to mission success. Agile principles that apply to deployment and sustainment include the following:

- Collaborate with users.
- Satisfy users through the delivery of useful software.
- Deliver working software frequently.

- Adapt to changing circumstances.
- Promote technical excellence and good design.

3.7.1 Collaborating with Users

Practices for **Collaborating with Users**

- **Expand the scope of the Development Team to include participation of site users and operators in preparing for implementation**
- **Provide a unified schedule for complete visibility of implementation activities**
- **Create a single, integrated bug tracking system for capturing and managing all issues**
- **Provide timely and effective training for users and operators**

An Agile development effort will have involved customer collaboration from inception. Business and Product Owners are integral to the Development Team, providing the system user and operator perspectives that ensure the operational suitability of delivered capabilities. Fulfilling those roles will typically involve coordination with many elements of the operating community that are or will be influenced by the products of Agile development. As implementation approaches, this coordination necessarily becomes more focused on the sites that will employ a capability first. Business and Product Owners should proactively involve site personnel in detailed implementation planning.

To provide coordination between the development, test, and operational communities, a unified authoritative schedule is necessary. The schedule should be readily accessible to all parties and updated regularly to ensure that it accurately reflects the status of implementation activities.

An Agile development effort will have an established backlog that includes identified software defects, the status of defect correction, and the priority of defect correction activity among all of the development tasks that need to be performed. Like the unified schedule, the backlog needs to be readily accessible to the operational community. As implementation approaches and a prospective operational software release is exposed to operational conditions in either operational testing or site integration activities, the pace of trouble reports may increase and observed problems may be more site-unique. It is important that site personnel be actively involved in the backlog management process and that the process be responsive to sites' priorities.

Although Agile values working software over documentation, the need for effective training is fundamental. Training may involve an approach such as user and operator manuals, but might also include more technology-oriented approaches such as computer-based training, help features incorporated with new capabilities, or training simulations that enable users and operators to gain familiarity with capabilities in offline or non-operational modes of the system. Like the operational capabilities, an Agile approach to training should involve solicitation and response to user feedback regarding training content, and iterative development of training materials responsive to user needs.

3.7.2 Satisfying Users through Delivery of Useful Software

Practices for Satisfying Users through Delivery of Useful Software

- **User stories and requirement prioritization promote utility of delivered software**
- **User feedback from iterative development (e.g., Sprint demos)**
- **Scope flexibility**
- **Measurements of use and post-op analysis**

An Agile development is successful when it satisfies customers by delivering software that users find useful. Agile promotes the likelihood of this outcome through application of Agile practices such as defining requirements through user stories and enforcing requirement prioritization through backlog management. User stories are grounded in the user perspective, so when implemented, they have a high probability of meeting user approval. The management of user-expressed priorities in the backlog ensures that development effort is directed to the capabilities of highest priority to the users. As implementation approaches, management of the backlog should increasingly represent the near-term priorities of the key sites and early implementers who will be most affected by new or modified capabilities.

Early and frequent user feedback is a fundamental feature of Agile development. At the completion of each Sprint, users have the opportunity to examine and respond to completed functions, ideally in the form of demonstrations of integrated capabilities. Delivery of operational capability to the users represents the final stage of this user feedback process, enabling user feedback on the integrated product in the operational environment.

Another fundamental aspect of Agile philosophy, scope flexibility, also promotes customer satisfaction. When adherence to the schedule discipline of Agile requires deferring functionality, prioritization of features in the backlog ensures that what the users find most valuable has been addressed first. Deferred developments should involve foregoing less useful features and enable on-time delivery of useful, albeit incomplete, software. Active and ongoing solicitation of priorities from initial implementers will ensure that their highest priority needs are being met.

In addition to subjective user feedback, the usefulness of delivered capabilities can be assessed through measurements of feature use and post-op analysis of effectiveness in achieving operational objectives. System analysis recording capabilities and offline data analysis can inform users and developers about the frequency and impact of feature use, and user surveys can provide subjective feedback of user perceptions. Consistent with this view, AMS promotes post-implementation reviews that are intended to verify the extent to which operational objectives and asserted benefits are realized.

3.7.3 Delivering Working Software Frequently

Practices for Delivering Working Software Frequently

- **Time-boxing in Sprints and releases**
- **Disciplined user story elicitation**

A fundamental property of Agile development is frequent delivery of working capabilities. This property is realized through the development and disciplined execution of Sprint and Release schedules that determine the pace of software development and delivery. However, the pace of development may not correspond to the pace of implementation. The pace of implementation must conform to the operational tempo of the environment in which the system or service resides. At one extreme, the environment might support the continuous integration involved with DevOps (e.g., frequently updated website content). At the other extreme, safety-critical operations such as those of the NAS will require rigorous OT&E of the integrated system and coordination of user and operator training prior to implementation (e.g., an En Route Automation Modernization release). These factors suggest a more episodic deployment based on the longer release schedule, similar to many existing NAS systems. When Sprint completions do not coincide with intended operational software releases, the pace of development can be maintained by delivering capabilities into a test environment in which assessment activities may be pursued in parallel with ongoing development.

Another property of Agile that promotes frequent delivery of working software is the disciplined definition of user stories. User stories should be defined at a level of granularity that enables their development within a Sprint. This user story discipline and the scope flexibility to defer development that does not fit within a Sprint ensures that useful features are delivered according to the Sprint/Release plan.

3.7.4 Adapting to Changing Circumstances

Practices for Adapting to Changing Circumstances

- **Backlog management**
- **User feedback**
- **Sprint and Release planning**

Changing circumstances can be expected in any complex operational environment. That is particularly true of the NAS. When a new capability is introduced to operation, the capability often needs to be modified to enhance operational suitability. Changes may involve either a general or site-specific level in response to weather, traffic conditions, anomalous events, etc. Responsiveness to change is a core value of Agile, and Agile practices are explicitly defined to support it. The backlog is expected to change with some regularity as requirements are added, deleted, or modified in response to such changing circumstances. Circumstances may suggest reordering priorities to address more pressing operational needs as they are encountered. User feedback during preparation for and execution of capability deployment will inform those decisions. The tempo dictated by Sprint and Release planning defines the opportunities to execute changes to the development plan.

3.7.5 Promoting Technical Excellence and Good Design

Practices for Promoting Technical Excellence and Good Design

- **Disciplined adherence to process**
- **Performance measurement**
- **Quality measurement**
- **Retrospectives**

Embracing change as described can have potential downsides. A large magnitude and high frequency of code changes can lead to lapses of technical standards adherence, loss of design coherence, and the accumulation of technical debt as development decisions favor expedience over design integrity. This risk is particularly manifest in preparation for or execution of deployment activities as schedule and operational needs come into potential conflict. This risk can be mitigated by maintaining a disciplined adherence to technical standards and to Agile practices such as effectively managing the backlog, continuing to time-box development and implementation activities, and maintaining a uniform pace of development. Technical excellence may also be preserved and promoted through effective measurement activities. Measurements may be defined to include both operational and technical performance metrics and metrics that more directly ensure that technical quality is maintained.

3.7.6 Agile Deployment Challenges

In practicing Agile deployment, specific challenges include the following:

- A basic tenet of Agile is scope flexibility, but there may be limits to this flexibility in the operational environment. The risk should be mitigated by effective backlog management, but there may still be situations when operational suitability demands meeting function and performance thresholds that are not negotiable. Examples might include replacing a legacy system or service that requires certain functions to be available, or implementing a safety-critical service with very high performance and design assurance requirements. These considerations should be reflected in a definition of the minimum viable product.
- The pace of acceptable change is determined by the operational context. While some FAA systems (e.g., many non-NAS systems) correspond to commercial counterparts that have demonstrated capacities for almost continuous change (e.g., DevOps), NAS operations involve safety- and efficiency-critical aspects that cannot be flexed to meet a schedule. In these situations, the frequent delivery of developed functions/features is made to a development/test baseline.

The development/test baseline is statused and introduced into operation at appropriate times determined by the operational context.

- User constituencies might not trust iterative development. An Agile procurement plan is likely to involve an iterative evolution of capability achieved by deployment of capabilities at a measured pace (e.g., six-month release cycle). Based on past experience, user communities might distrust a plan that fails to meet all needs initially, promising future capability growth. There are many examples in government procurement of promised enhancements failing to be realized. One approach to addressing this concern is to define the minimum viable product to include the "must haves" identified by the user community.
- Integration in the operational environment is always challenging, particularly for new systems that might involve numerous complex interfaces. In these situations, Agile may actually be an advantage, driven by prioritization of the work to be done, time-boxing, and proceeding at an established development pace.
- A documentation focus is deprecated in Agile development, but the quality of training documentation can be fundamental to implementation success. The quality of user manuals, operating instructions, installation and test scripts, etc., strongly influences the user perception of the system or service and its effective use. One approach is to evolve the training documentation concurrent with the product development, effectively viewing it as an Agile product.

3.8 Agile Program Management

The FAA defines program management as establishing clear and achievable objectives; balancing the competing demands for quality, scope, time, and cost; and developing an approach to address the different stakeholder concerns [43]. In a broader sense, program management provides the necessary organization and administration to accomplish smooth development, deployment, maintenance, and end-of-life activities. Given this, program management is necessary across the entirety of an acquisition's lifecycle, and is responsible for monitoring and control of three specific elements: cost, schedule, and scope. Facets of program management typically include the following:

- Program formulation and planning
- Project monitoring and control
- Stakeholder coordination (e.g., customer interaction, contractor [primes or subs] management)
- Risk management.

Artifacts typically associated with these processes include work breakdown structures and SOWs for program scope definition, integrated master schedules for schedule control and monitoring, and Earned Value Management (EVM) for performance assessment and tracking.

Many of these topics are covered in detail throughout this document and will not be repeated here; please refer to Sections 3.1, 3.2, 3.4, and 3.5 for more information regarding team construction, planning, estimating, and contracting an Agile Acquisition, respectively. The remainder of this section will expand on the evolution of program management for Agile Acquisitions and where challenges remain.

In reviewing the Agile principles provided in Table 1 1, a case can be made that program management must evolve to consider and incorporate facets of each of the 12 principles. In particular, the following Agile principles must be applied to program management to update typical activities:

- Close, daily collaboration between business people and developers.
- Building projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
- Self-organizing teams.
- Simplicity—the art of maximizing the amount of work not done.
- Regular adaptation to changing circumstances.

The first three principles can be loosely grouped into a single topic, fostering a high-performance cross-organizational team. The final two principles operate together for effective technical planning, monitoring, and execution. The following sections provide Agile practices for these two topics.

3.8.1 Fostering a High-Performance Cross-Organizational Team

Practices for Fostering a High-Performance Cross-Organizational Team

- **Provide the overall vision**
- **Identify qualified, motivated individuals who will be committed to mission goals**
- **Instantiate self-organizing teams and provide guidance on expectations, technically and programmatically, and allow them to operate with relative autonomy**
- **Facilitate shared team vision and guidelines for close cooperation and collaboration among all team members**
- **Listen to team member needs and enable the removal of obstacles as necessary**
- **Connect enterprise, program, and team-level personnel by providing upward and downward management and creating opportunities for information and technical exchange**

In a traditional acquisition, the activities of program management are typically performed by an individual, the program manager, or designees. In an acquisition where Agile practices have been adopted, the responsibility for program management becomes blurred, and the entire team begins to shoulder pieces of this responsibility. Product Owners collaborate with developers and customers to facilitate effective program formulation and planning, development of requirements, Sprint definitions, and creation of backlogs. Scrum Masters, if employed, remove barriers to optimum team functioning, keeping the entire team progressing and focused on prioritized tasks at hand.

The following practices are recommended for fostering a high-performance cross-organizational team with effective Agile program management:

- Provide the overall vision. As specified in Section 3.2, for Agile planning, it is critical to have a clear understanding of the overall vision and how the program is to meet mission (e.g., customer) needs. This big picture provides a foundation on which lower level tasks can be evaluated for value and prioritized to meet the larger vision expediently. Vision needs to be provided on multiple levels, starting with the enterprise-wide big picture, to the program level and how each program fits within the enterprise, down to the project and task levels. This will keep all teams informed and able to adapt and respond quickly to necessary changes.
- Identify qualified, motivated individuals who will be committed to mission goals. Due to the constrained iteration timeframes required by Agile, Development Teams need to be constructed to minimize learning curve timelines and external distractions. This is not to say that junior team members or qualified individuals with other project obligations are incompatible with Agile. Training and planning for resource availability must be included in project planning to ensure high productivity.
- Instantiate self-organizing teams and provide guidance on expectations, technically and programmatically, and allow them to operate with relative autonomy. A high-performing team, when given the responsibility and authority to enact necessary changes, can make informed and rational decisions. Providing guidelines and expectations enables the team to recognize and prioritize items to best fit the team's working style. Furthermore, by allowing the Development Team to operate with autonomy, the program can push decision making down to the most effective level, enabling proactive changes rather than reactive decisions, which may or may not be the most fully informed.
- Facilitate shared team vision and guidelines for close cooperation and collaboration among all team members. It is imperative that program management make sure all team members are on the same page. Without this vision and guidance, team members may work at odds with each other and eliminate the gains that may be achieved with Agile and adaptive processes. Program management aligns team members, clarifies areas of confusion, and smooths the way for development and execution of the program.
- Listen to team member needs and enable the removal of obstacles as necessary. Similar to traditional program management processes, resource management remains key in executing an Agile program. Ensuring that members of the Development Team have access to stakeholders and tools, the schedule to complete priority items, and the budget to achieve is necessary for success.
- Connect enterprise, program, and team-level personnel by providing upward and downward management and creating opportunities for information and technical exchange. This connection and information exchange also needs to occur with other teams developing or sustaining solutions. This open communication will ensure that integration is maximized and interfaces are leveraged, reducing the potential for overlap and duplication. With aggressive schedules, open and transparent information exchange is required to maintain progress. If developers are unaware of program vision shifts, they cannot make adjustments to Sprint formulation. If enterprise personnel are unaware of challenges within development, they cannot suggest alterations and prioritizations of mission needs. Upward management to the enterprise level and downward management to

the development level are necessary to make sure these lines of communication are open and efficiency is preserved and even improved.

3.8.2 Providing Effective Technical Planning, Monitoring, and Execution with Agile Program Management

Practices for Providing Effective Technical Planning, Monitoring, and Execution with Agile Program Management

- **Assimilate inputs and feedback from all team members to balance technical and programmatic needs with business value**
- **Track and monitor program/project/task progress through a variety of methods rather than traditional cost/schedule means**
- **Realign team priorities as necessary in conjunction with leads and Product Owners**
- **Foster the identification of lessons learned and best practices and fold them back into program execution**

Change is necessary and fundamental to an Agile Acquisition to provide maximum value within the shortest time span feasible. Program management must enable flexibility and take commonsense approaches to technical planning, monitoring, and execution activities. Program management should generate metrics (e.g., expected and achieved velocities, backlog burn-down, defects) as appropriate to understand impacts of change on cost, schedule, and scope baselines. Once impacts are identified, embrace adaptive methodologies to adjust these baselines to maintain the most efficient and direct path to achieving the overall program vision. The following practices are recommended for effective technical planning, monitoring, and execution with Agile program management:

- Assimilate inputs and feedback from all team members to balance technical and programmatic needs with business value. Just as with program management in traditional acquisitions, those with program management responsibilities must listen to and combine status, estimations, and other pieces of information from stakeholders and all levels of the team. The difference and the consideration for Agile practice is to identify change to scope rather than change to schedule or budget. It should be noted that change to scope is not meant to be carte blanche to defer or reduce capabilities to ease development workload, but rather a means to achieve maximum value given technical challenges and resource considerations. An initial scope, or minimum viable product, must be defined. Once defined and agreed upon, the need to balance technical and resource solutions with value should be considered continuously throughout the project, not just at typical specified intervals.
- Track and monitor program/project/task progress through a variety of methods rather than traditional cost/schedule means. This is a program management practice that begins to

significantly diverge from traditional practices. In traditional practice, progress is monitored against a plan defined at the beginning of the program. Changes can be made, but they are geared toward schedule and budget changes rather than scope because value is measured by completion of defined artifacts (e.g., EVM). With Agile activities, variance to a plan does not provide a good metric because the plan is meant to be adaptable, allowed to shift and change as necessary to best accomplish the mission goals. New metrics and methods of collecting status and measuring progress toward mission success are required.

- Realign team priorities as necessary in conjunction with leads and Product Owners. As discussed in Section 3.1.3, for Agile teams, dedicated (i.e., allocated) individuals are necessary to achieve aggressive schedules. Program management should identify and remove obstacles to staff planning and allocation, ensuring that staff have enough time and availability to accomplish program goals. Also, program management with its monitoring of progress should identify when teams begin to deviate from planned tasking and affect changes to the plan or staff direction as necessary.
- Foster the identification of lessons learned and best practices and fold them back into program execution. This practice is not unique to Agile; however, the value of having discussions, retrospectives, and evaluating program performance for these items cannot be overstressed. Agile allows for rapid course correction due to its iterative nature, and incorporating this traditional program management practice allows for continuous performance improvement.

3.8.3 Agile Program Management Challenges

While the practices to accomplish traditional and Agile program management may differ, the goals of program management and Agile principles are very compatible and complementary. Each requires the following:

- Project/task formulation with emphasis on effective planning
- Customer engagement to ensure that mission needs are met satisfactorily
- Monitoring and control of progress to understand when adjustments are necessary
- Identification and mitigation of risks to maintain and accelerate project progress.

Challenges remain in the following two areas in applying the Agile principle of providing effective technical planning, monitoring, and execution with Agile program management:

- Contractor/subcontractor management: When the execution of Agile strategies is wholly contained within one organization, responsibilities can be easily understood and assumed or designated. When two or more organizations are involved, much more of a coordination burden needs to be tackled to ensure that each stakeholder/responsible entity understands and agrees to the evolving scope. This challenge should primarily be addressed through Agile contracting strategies with clear guidelines for expectations and continuous communication and collaboration.
- Monitoring of program progress: Depending on program cost, current federal regulations require the use of an Earned Value Management System [44], [45], [46]. For FAA acquisitions, EVM requirements are determined by the EVM Focal Point and the JRC, and they can be applied at both the program and developer levels. There is a known difficulty in connecting Agile to

these traditional accounting and monitoring structures because EVM typically requires a strong understanding of all of the needs and plans up front in significant detail. EVM monitors programs for value (e.g., cost and schedule) by claiming credit when completing intermediate products as defined by the up front plans as the program progresses, evaluating and addressing variances between actual and planned costs and schedule [47]. The lack of flexibility in EVM processes to account for incremental and adaptive planning and the difficulties associated with applying the monitoring method to the outputs of an Agile program (e.g., user stories versus traditional documentation) have been identified by a number of organizations [48], [12]. Variances from the cost and schedule baselines are to be expected with the Agile principle of welcoming changing requirements, so new methodologies are required for monitoring program progress. Development and investigation of these new methods is ongoing [48], [49].

3.9 Enterprise Architecture—Transition to Agile

For the federal agencies, including the FAA, an enterprise architecture (EA) establishes the agency-wide roadmaps that capture the plans for achieving the agency's missions through optimal performance (better, faster, and cheaper) of its core business processes.

Note that the term "enterprise architecture" may take on different meanings when being discussed at the enterprise level and at the program level. At the enterprise level, the EA provides a systematic way of capturing the information that helps to create a blueprint for the agency's current and the desired/target state. One of its primary uses is to help inform the external stakeholder such as OMB about the agency's management of resources and progress. An EA acts as a common language for business and IT. It is a means to address crosscutting concerns/risks, such as integration and interoperability concerns, before they manifest themselves. At the program level, the term EA may be used in place of "architecture" to describe program-specific key elements, relationships, and architectural viewpoints. This document makes that distinction by using either EA or program-level architecture, as appropriate.

The FAA's NAS EA provides users with the agency's multiyear strategic plan and framework for improving and evolving the NAS from the current portfolio of fielded ATM services and capabilities through the next 10- to 15-year planning cycle [50]. The FAA's "non-NAS" EA captures information and plans related to use of IT for evolving and improving its mission support capabilities and services, such as its Human Resource Information System, and Personnel and Payroll functions.

The Agile principles that apply to EA development and management and equally to program-level architecture development include the following:

- Valuing individuals and interactions over processes and tools
- Maintaining simplicity—the art of developing "good enough"
- Engaging stakeholders continuously
- Adapting to changing circumstances.

3.9.1 Valuing Individuals and Interactions over Processes and Tools

Practices for Valuing Individuals and Interactions over Processes and Tools

- **Architecture views and metrics take precedence over documents/reports**
- **Enterprise architects drive collaboration of programs and teams around a common technical vision**
- **Enterprise architects maintain personal connections with each Agile initiative (Agile Release Train) [31]**
- **Enterprise architects participate in program-level design and demonstrations**
- **Teams provide enterprise architects with full visibility into practices and challenges**

Common problems that large organizations experience with respect to EA include the following:

- EA is outdated/obsolete by the time it is published.
- There is general unawareness about enterprise-level architecture efforts.
- Teams have misconceptions about the organization's vision for the target state.
- Teams do not adhere to the EA standards, design patterns, and best practices.
- Teams are not aware of changes to the EA.
- There is a general perception that EA views and products development are time-consuming.
- Teams do not see value in EA products.

A theme running through organizations that experience these problems is a culture of command and control, where emphasis is on products, processes, and tools over individuals and interactions.

It is more important to have an EA that stakeholders at varying levels of the organization know about, actively participate in ensuring that it is up-to-date, and take advantage of. Trying to build the perfect architecture and answer all questions related to modeling techniques, standard notations, and tools is not feasible. The Agile approach advocates focus on the architecture views, and clear metrics associated with those views, instead of generating comprehensive reports and documentation that may be out-of-date by the publication date.

In an Agile organization, the focus is on people and interactions at the varying levels of the organization. In order to facilitate close collaboration and interaction within the organization, the role of architect is typically divided into two—the enterprise architect and the system architect.

The enterprise architect(s) and enterprise resources are available to the programs and Agile Teams, as well as to the public (e.g., an EA portal with an integrated dashboard), if appropriate. The enterprise architect operates across programs, and is responsible for understanding the overarching organization's mission, the as-is state, and the target state. The enterprise architect is responsible for ensuring that the IT development strategies and technologies are aligned with the organization's evolving business needs.

More specifically, the enterprise architect is responsible for the following:

- Work collaboratively with business stakeholders (portfolio managers, users or user communities) as well as the system architects around a common vision and understanding of the end state
- Maintain personal connections with each Agile initiative, and participate in program-level architecture decisions
- Inform and influence the common modeling, notation, design patterns, and standards used
- Keep abreast of innovations that can have a crosscutting effect on the enterprise
- Monitor trends in industry and government, and apply innovative solutions to EA challenges
- Identify architectural or technical shortfalls (e.g., security, and scalability).

The system architect role is similar in function to that of the enterprise architect, with the primary difference that system architects operate at the program level and have a much more intimate relationship with product managers and the Agile Teams. The system architect's basic function is to:

- Maintain an understanding of the expressed needs and direction at the enterprise level, in close collaboration with the enterprise architect
- Evolve and maintain the system's architecture and implementation framework for current and upcoming needs in an iterative manner
- Ensure that there is transparency with respect to practices and challenges that the program maybe facing.

Often, the role of the system architect is carried out by the lead engineer.

3.9.2 Maintaining Simplicity—the Art of Developing "Good Enough"

Practices for Maintaining Simplicity—the Art of Developing "Good Enough"

- **EA models do not need to be done to perfection and should be kept simple**
- **Automated tools and whiteboards may be used to capture EA inputs**

The Agile approach suggests a "good enough" philosophy, which should be the same philosophy used when developing architecture models. These models do not need to be done well in advance or done to perfection, and need to be kept simple.

At the Sprint level, the first iteration of an architecture view may be done on a whiteboard or in a non-traditional architecture tool to keep things simple. The initial architecture view should help expedite communication and resolution discovery by the Development Team and the system architect.

At the enterprise level, the enterprise architect should inform all stakeholders of the upcoming changes and their possible impact on the architecture. In many organizations, including the FAA in the past, the EA is developed and published on an annual cycle. As a result, new information and architectural guidance available at the beginning of the year is not included in the guide until the next publication of the EA products. A more frequent update cycle, communication of changes, and timely guidance are often far better than speculation at the Agile Teams' level, even if that guidance is still not fully matured.

To be more Agile, the FAA and organizations alike are employing tools to help with more rapid update cycles and increased availability of enterprise-level views and information. Utilizing automated tools frees up resources that would otherwise be occupied developing the EA products to perform analysis on the EA products.

3.9.3 Engaging Stakeholders Continuously

Practices for Engaging Stakeholders Continuously

- **Enterprise architects work with business stakeholders and system architects**
- **Enterprise architects must speak their customers' language**
- **Enterprise architects have to actively look for opportunities to remove waste**

An Agile organization is functionally Agile when its operational parts collaborate, and anticipate and embrace change. Enterprise architect(s) need to work with their customers, often the business stakeholders (program managers and/or user community) and system architects in a manner that is most effective and efficient. This means that communication with the customers may need to take form of teleconferences, video conferences, and if need be face-to-face meetings.

The enterprise architect can be the connecting role between the stakeholders with the long-term vision and the project teams. So, it is important for the enterprise architect to maintain good working relationships with those stakeholders.

Enterprise architects are key participants in helping to define, maintain, and communicate strategic themes and key business drivers, and reporting on progress toward an envisioned end state. Tools such as electronic mail, portals, and dashboards are used to reach the community of stakeholders.

At the program level, the system architect is engaged in helping define, analyze, and refine functional and non-functional requirements using operational architecture views. It is important that the architecture and requirements work seamlessly and continuously collaborate to ensure that requirements are aligned with the target state defined in the EA, and are technically achievable.

3.9.4 Adapting to Changing Circumstances

Practices for Adapting to Changing Circumstances

- **Develop architecture models more iteratively**
- **Identify issues early through shorter iterations**
- **Continually refine architecture and technical strategies to adapt to the needs**
- **Use EA to help reduce the cycle time for fielding capabilities**

The Agile approach works well when dealing with small-scale applications that are being developed or integrated into an existing operational baseline or system. For large-scale IT solutions, the Agile approach needs to have architecture that is developed ahead of the development Sprints. Future refinement of the architecture can be iterative and alongside the development Sprints.

Shorter cycles help focus on current values/needs. They allow the enterprise to adapt to needs, advancements in technology, and shifting organizational priorities. EA models need to be developed iteratively, and when ready, smaller pieces will be linked together to create greater wholes.

The primary reason for not producing a comprehensive set of documents and models up front is that detailed artifacts are often ignored by the Development Teams, and become shelf ware. Instead, the focus under an Agile approach should be on EA products of recognizable value to all stakeholders. In addition, the artifacts that are developed should focus on architectural principles and standards. This will help to make sure that program managers, Product Owner, and Development Teams understand those principles and standards, and that once a clear architecture is in place, they all can continue to contribute toward refining it.

Beyond the development process, the EA itself can be used to identify and enforce technologies that facilitate Agile engineering practices. It can also be used to identify solution components that can be reused across the enterprise, which will reduce the cycle time for fielding capability to users for feedback.

3.9.5 Challenges with Applying Agile to Enterprise Architecture

The Agile approach embraces a philosophy that all requirements cannot be known in advance. This trait helps organizations with rapidly evolving IT environments continue to operate and function effectively and not fall behind the technology evolution by being stuck in the planning and definition phase. Given that EA is primarily focused on up front risk reduction, it seems that Agile philosophy and EA may be in direct conflict. For example, Agile advocates less emphasis on documentation, whereas the current approach to EA relies on documentation as a primary means of communication. EA requires lead time for development, and eventually analysis, whereas Agile is advocating a much more iterative development and analysis approach. However, the incompatibility of Agile and EA may be skin-deep, as once Agile principles are closely examined, they can be adopted and help modify how EA management and development is approached, and how that can become the operating norm for an organization like the FAA.

One primary area that the Agile approach does not address clearly and for which further research is necessary is the fact that the techniques do not include an explicit method of ensuring agency-wide compliance with the EA. In other words, Agile places great emphasis on the human element, as it depends on people being responsible, striving for excellence, and collaborating often and with the right stakeholders. Agile application at the program and Release levels is well understood, but maintaining program- and project-level visions in synchronization with the enterprise level is more challenging. Both policy and practices need to evolve, be developed, and be put in practice to help achieve that goal.

NEXT STEPS

4

The FAA is currently in the early phases of adopting Agile, and this document has provided the key principles and practices that should be considered as the FAA integrates this new approach. Government agencies have increasingly leveraged Agile practices in recent years. The increased application of Agile methods across the federal government combined with the increasing number of Agile initiatives within the FAA make Agile Acquisition an important area to address. The following steps are needed for continued progress in Agile adoption in FAA acquisition efforts.

- The FAA's leadership should make key decisions related to the implications of Agile Acquisition on the AMS process. The decisions include whether Agile implies the need for a suitable ACAT versus the need to reconsider existing AMS steps more broadly. They also include determining whether to reconsider the criteria for certain AMS artifacts, among other issues.
- Based on the key decisions made, the FAA should begin implementation planning for Agile. This includes conducting an organizational scan to assess readiness and challenges, and identifying potential champions for the approach, as well as key stakeholders and decision makers. It also includes developing a roadmap for Agile implementation, executing the steps identified in the roadmap, and refining the process through iterative feedback.
- Identifying an Agile Pilot program is an important step. The program should be selected on the basis of the criteria described in this document, in conjunction with evaluating the business motivations of the program. The Agile Pilot serves as a basis for the validation of the principles and practices identified in this document. It also serves as a mechanism to provide feedback into and further mature FAA's Agile Acquisition body of knowledge. Tabletop exercises can provide a valuable means to assess the potential implications of Agile adoption before a program is selected.
- Developing the next iteration of these principles and practices, including examples of tailored Agile Acquisition approaches, is also necessary. It is important to identify the steps and key artifacts needed for an Agile program within the context of the AMS. Adopting an Agile paradigm implies changes to the process and artifacts typically required of the AMS process, including the content of those artifacts. Section 2.3 suggests an Agile lifecycle alternative; however, there is still a need to develop concurrence on an Agile lifecycle model that is specific to the FAA environment. The Agile Pilot can serve as a basis for identifying and validating this model and artifacts.
- Educating the FAA's workforce on the Agile principles and practices described in this document is another essential step. Training needs to be conducted across all the key acquisition organizations in the FAA's enterprise. Through the education sessions, issues may be identified that can be used as a basis to further mature the body of knowledge in Agile Acquisition.
- Finally, in conjunction with the work on Agile Acquisition, it is important to investigate the applicability of emergent Agile systems engineering practices on the FAA. Acquisition and systems

engineering are very synergistic within the FAA, and it is important for each of these areas to mature if the FAA is to truly embrace an Agile paradigm. A key factor to consider is the safety-critical nature of air traffic control systems, and the implications of Agile systems engineering practices on these systems. The FAA should consider leveraging work currently being pursued by INCOSE, in addition to the work that has matured within other agencies, to assess the implications for its current systems engineering practices.

REFERENCES

5

[1] The Standish Group, "CHAOS 2014," The Standish Group International, Inc., 2015. [Online]. Available: http://www.lulu.com/shop/james-johnson/chaos-2014/paperback/product-22302662.html. [Accessed 16 September 2015].

[2] "Manifesto for Agile Software Development," Ward Cunningham, 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 10 September 2015].

[3] S. Ambler, "Disciplined Agile 2.0: A Process Decision Framework for Enterprise I.T.," [Online]. Available: http://www.disciplinedagiledelivery.com/. [Accessed 13 October 2015].

[4] C. Larman and B. Vodde, "Scaling Scrum, Scaled Agile - Large Scale Scrum (LeSS)," [Online]. Available: http://less.works/. [Accessed 13 October 2015].

[5] Scaled Agile Inc., "Scaled Agile Framework," Scaled Agile Inc., [Online]. Available: http://www.scaledagileframework.com/. [Accessed 16 September 2015].

[6] Wikipedia, "Scrum (Software Development)," [Online]. Available: https://en.wikipedia.org/wiki/Scrum_%28software_development%29. [Accessed 13 October 2015].

[7] S. C. Pete Modigliani, "Defense Agile Acquisition Guide," The MITRE Corporation, McLean, VA, 2014.

[8] International Council On Systems Engineering (INCOSE), "INCOSE Systems Engineering Handbook," International Council On Systems Engineering, 29 June 2015. [Online]. Available: http://sebokwiki.org/wiki/INCOSE_Systems_Engineering_Handbook. [Accessed 16 September 2015].

[9] "U.S. Digital Services Playbook," The White House, [Online]. Available: https://playbook.cio.gov/. [Accessed 16 September 2015].

[10] "The TechFAR Handbook," [Online]. Available: https://playbook.cio.gov/techfar/. [Accessed 16 September 2015].

[11] Defense Information Systems Agency (DISA), "ABOUT GCSS-J," Defense Information Systems Agency (DISA), [Online]. Available: http://disa.mil/Mission-Support/Command-and-Control/GCSS-J/About. [Accessed 16 September 2015].

[12] Government Accountability Office, "Effective Practices and Federal Challenges in Applying Agile Methods - GAO-12-681," Government Accountability Office, 2012.

[13] Scrum Inc., "The Scrum Guide," Scrum.org, [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed 17 September 2015].

[14] American Psychological Association, "Multitasking: Switching costs," American Psychological Association, 20 March 2006. [Online]. Available: http://www.apa.org/research/action/multitask.aspx. [Accessed 10 September 2015].

[15] G. A. Miller, "The magical number seven, plus or minus two: some limits on our capacity for processing information," Psychological review, vol. 63, no. 2, pp. 343-352 , 1956.

[16] R. Dunbar, "Neocortex Size as a Constraint on Group Size in Primates," *Journal of Human Evolution*, vol. 22, no. 6, pp. 469-493, 1992.

[17] S. Raman, "Five Levels of Agile Planning," Scrum Alliance: Agile Atlas, 15 January 2014. [Online]. Available: http://agileatlas.org/articles/item/five-levels-of-agile-planning. [Accessed 17 September 2015].

[18] Scrum Training Series, “Module 3: Sprint Planning Meeting,” Collabnet, [Online]. Available: http://scrumtrainingseries.com/SprintPlanningMeeting/SprintPlanningMeeting.htm. [Accessed 17 September 2015].

[19] C. Northern et al., Handbook for Implementing an Agile Systems Engineering Process in the Department of Defense, McLean, VA: The MITRE Corporation, 2010.

[20] J. Gariano, “Hardened Agility: Deploying Agile at a Defense Contractor,” General Dynamics, Scottsdale, AZ, 2015.

[21] D. Leffingwell, “Scaled Agile Framework - Epics,” Scaled Agile, Inc, 25 July 2014. [Online]. Available: http://www.scaledagileframework.com/epics/. [Accessed 18 September 2015].

[22] N. Subowo, “Towards an Agile Systems Engineering Approach for the National Airspace System,” in SEDC 2014, Washington, DC, 2014.

[23] W. A. Huckabee, “Requirements Engineering inn an Agile Software Development Environment,” Defense ARJ, vol. 22, no. 4, pp. 394-415, October 2015.

[24] B. Cao and L. Ramesh, “Agile Requirements Engineering Practices: An Empirical Study,” IEEE Software, pp. 60-67, 2008.

[25] M. Lant, “How to Easily Prioritize Your Agile Stories,” 21 May 2010. [Online]. Available: http://michaellant.com/2010/05/21/how-to-easily-prioritize-your-agile-stories/. [Accessed September 2015].

[26] D. Leffingwell, “Weighted Shortest Job First (WSJF) Abstract,” Scaled Agile Framework, July 2014. [Online]. Available: http://www.scaledagileframework.com/wsjf/. [Accessed September 2015].

[27] G. Sillitti and A. Succi, “Requirements Engineering for Agile Methods,” in Engineering and Managing Software Requirements, Heidelberg, Germany, Springer Berlin, 2005, pp. 309-326.

[28] S. W. Ambler, “Agile Requirements Change Management,” Ambysoft Inc., [Online]. Available: http://agilemodeling.com/essays/changeManagement.htm. [Accessed 18 September 2015].

[29] Government Accountability Office, GAO Cost Estimating and Assessment Guide; Best Practices for Developing and Managing Capital Program Costs GAO-09-3SP, Washington, DC: GAO, 2009.

[30] M. Cohn, Agile Estimating and Planning, 2006.

[31] National Defense Industrial Association, “Guide to Earned Value Management for Agile Programs: Industry Best Practice. Draft Version 1.0,” NDIA, Arlington, VA, 2015.

[32] D. Leffingwell , “Agile Release Train Abstract,” Scaled Agile, Inc., 22 July 2014. [Online]. Available: http://scaledagileframework.com/agile-release-train/. [Accessed 22 October 2015].

[33] International Cost Estimating and Analysis Association, Cost Estimating Body of Knowledge. Version 1.2. Chapter 4: Data Collection and Normalization., Vienna, VA: ICEAA, 2012.

[34] Federal Aviation Administration, “FAA Acquisition System Toolset (FAST),” Federal Aviation Administration, July 2015. [Online]. Available: http://fast.faa.gov/Index.cfm?p_title=Fast Home. [Accessed 18 September 2015].

[35] E. Gross et al., “Contracting for Agile Software Development in the Department of Defense: An Introduction (CMU/SEI-2015-TN-006),” Software Engineering Institute (SEI), Pittsburgh, PA, 2015.

[36] Harris Corporation, “Harris Corporation Awarded $331 Million Contract by FAA for Data Communications Integrated Services Program,” Harris Corporation, 20 September 2012. [Online]. Available: http://harris.com/view_pressrelease.asp?pr_id=3518. [Accessed 21 September 2015].

[37] Office of Management and Budget, “Contracting Guidance to Support Modular Development,” Office of Management and Budget, Washington, DC, 2012.

[38] Defense Acquisition University , “Test and Evaluation Community of Practice (T&E CoP),” [Online]. Available: https://acc.dau.mil/CommunityBrowser.aspx?id=18011&lang=en-US. [Accessed 15 September 2015].

[39] Federal Aviation Administration, “Acquisition Management Policy - 4.4 Test and Evaluation,” Federal Aviation Administration, July 2015. [Online]. Available: http://fast.faa.gov/docs/acquisitionManagementPolicy/AcquisitionManagementPolicy4.4.pdf#nameddest=policy4_4. [Accessed 15 September 2015].

[40] Federal Aviation Administration , “Acquisition Management Policy - 4.5 Independent Operational Assessment,” July 2015. [Online]. Available: http://fast.faa.gov/docs/acquisitionManagementPolicy/AcquisitionManagementPolicy4.5.pdf. [Accessed 15 September 2015].

[41] Federal Aviation Administration , Test and Evaluation Process Guidelines, Washington, DC: Federal Aviation Administration, 2014.

[42] “Test and Evaluation Handbook, Version 3.0,” FAA William J. Hughes Technical Center, Atlantic City International Airport, 2013.

[43] Federal Aviation Adminstration, “Program Management Practice Toolkit,” FAA Acquisition Progessions Online Community, [Online]. Available: https://ksn2.faa.gov/faa/AcquisitionProfessions/Practices/Pages/Disc_PM.aspx. [Accessed October 2015].

[44] “Subpart 34.2-Earned Value Management System,” March 2005. [Online]. Available: https://www.acquisition.gov/sites/default/files/current/far/html/Subpart%2034_2.html. [Accessed October 2015].

[45] Office of Management and Budget, “CIRCULAR NO. A–11 Perparation, Submission, and Execution of the Budget,” Office of Management and Budget, Washington, DC, 2015.

[46] Federal Aviation Adminstration, “Acquisition Management Policy 4.16 Earned Value Management,” Federal Aviation Adminstration, Washington, DC, 2015.

[47] GEIA, “Earned Value Management Systems ANSI/EIA-7-48-B-2007,” Government Electronics and Information Technology Association , Arlington, VA, 2007.

[48] M. A. Lapham, et. al, “Agile Methods: Selected DoD Management and Acquisition Concerns CMU/SEI-2011-TN-002,” Software Engineering Institute, Hanscom AFB, MA, 2011.

[49] G. Kranz, “EVM and AGILE in a DoD Environment,” Perfomance Assessments and Root Cause Analysis (PARCA) EVM Webinar, June 2015.

[50] Federal Aviation Adminstration, “NAS Integrated Systems Engineering Framework (ISEF), version 3.3,” Federal Aviation Administration, Washington, DC, 2014.

[51] Federal Aviation Administration, “Acquisition Management Policy,” July 2015. [Online]. Available: http://fast.faa.gov/docs/acquisitionManagementPolicy/acquisitionManagementPolicy.pdf. [Accessed 18 September 2015].

[52] Federal Aviation Administration, “AMS Policy vs Guidance,” Federal Aviation Administration, [Online]. Available: http://fast.faa.gov/AMSPolicyVsGuidance.cfm?p_title=AMS Information. [Accessed 18 September 2015].

[53] Federal Aviation Administration, “AMS Tailoring,” Federal Aviation Administration, [Online]. Available: http://fast.faa.gov/AMSTailoring.cfm?p_title=AMS%20Tailoring. [Accessed 18 September 2015].

[54] Federal Aviation Administration , “AMS Tailoring Request Process,” [Online]. Available: http://fast.faa.gov/docs/AMSTailoringRequestProcess.doc. [Accessed 18 September 2015].

[55] Federal Aviation Administration, “AMS Tailoring Request Template,” [Online]. Available: http://fast.faa.gov/docs/AMSTailoringRequestTemplate.doc. [Accessed 18 September 2015].

[56] Federal Aviation Administration, “AMS Table of Acquisition Categories,” [Online]. Available: http://fast.faa.gov/docs/acqcattable.doc. [Accessed 18 September 2015].

[57] T. Sulaiman, B. Barton, T. Blackburn, “AgileEVM—Earned Value Management in Scrum Projects,” in Proceedings of Agile 2006, Minneapolis, MN, 2006.

APPENDIX A
GLOSSARY AND LIST OF ABBREVIATIONS

A.1 Glossary

Term	Definition
Agile	The ability to respond efficiently and effectively to changes in the operational environment. Efficiency and effectiveness refer to being timely, affordable, and useful.
Agile Acquisition	Strategy for providing multiple, rapid deliveries of incremental capabilities for operational use and evaluation. It is a combination of Agile principles, acquisition policies, and Program Office operations that will enable the rapid delivery of services and systems.
Backlog	Repository for all of the upcoming work needed to satisfy the program's solution (the objective). There are different levels of a backlog: program, Release, and Sprint.
	Program backlog—Primary source of all requirements/desired functionality for the program. Release backlog—Subset of the program backlog listing features intended for the release. Sprint backlog—Subset of the Release backlog listing the user stories to implement in the Sprint. [7]
Burn-Down Chart	Graphical representation of work left to do versus time.
Cadence	Development at a sustainable pace, where there is a consistent flow of activities (e.g., planning and deployment and retrospective) within the time-box.
Lead Engineer	Oversees technical aspects of the program, working with architects and systems engineers to ensure that the system is developed in alignment with FAA's evolving enterprise architecture and technical standards.
Daily Commitment (Daily Scrum)	Team synchronization meeting (typically 15 to 30 minutes) to plan activities and assess progress and impediments. [7]
Definition of Done	Agreeing on a clear understanding of what it means for a user story or piece of functionality to be complete, or for a product increment to be ready for release. [7] It is from the perspective of the project, and is not to be confused with "Acceptance Criteria," which is to prove that a user story is complete and meets the user's purpose.
Development Team	A group (ideally 7 +/- 2 members) responsible for conducting the development work necessary to deliver a potentially releasable increment of functionality upon completion of each Sprint.

Term	Definition
Development Team	A group (ideally 7 +/- 2 members) responsible for conducting the development work necessary to deliver a potentially releasable increment of functionality upon completion of each Sprint.
DevOps	Development practice where the users, systems engineers, and Development Team work in close collaborative environment throughout the lifecycle.
Epic	A large user story, often defined for a release that spans multiple Sprints. [7]
	Enterprise initiatives that are sufficiently substantial in scope. There are two different types of epics: business, which are large customer-facing initiatives that encapsulate the new development necessary to realize the benefits of some new business opportunities, and architectural, which are large technology initiatives necessary to evolve portfolio solutions in order to support current and future business needs.
Features	Services provided by the system that fulfill one or more stakeholder needs. They are maintained in the program backlog and are sized to fit in a Release.
Minimum Viable Product	Minimum set of features that will bring value to the user community
Product Owner	FAA representative of the user community who is responsible for maximizing the value of the work conducted by the Development Team.
Program Manager	Oversees the planning and execution of the program, ensures adherence to FAA Agile principles and practices, and represents the key interface between enterprise and Development Team levels.
Scrum	Framework within which people can address complex adaptive problems while productively and creatively delivering products of the highest possible value.
Scrum Master	The developer who is responsible for maintaining the structure and the Scrum rules of operation. Within the FAA environment, the Scrum Master may be a contractor.
Sprint	A time-boxed period during which potentially releasable "working capability" is developed.
Story Points	Unit of measurement to estimate relative complexity of user stories. [7]
Release	Capability delivered to users, composed of multiple Sprints. [7]
User Stories	Sprint-level requirements expressed as a description of the functionality desired by a user, usually represented in a structured format and managed in Sprint backlogs.
Velocity	Rate at which functionality is being delivered (e.g., story points/Sprint).

A.2 List of Abbreviations

Acronym	Definition
ACAT	Acquisition Category
AEB	Acquisition Executive Board
AMS	Acquisition Management System
ATM	Air Traffic Management
CO	Contracting Officer
DAD	Disciplined Agile Delivery
DoD	Department of Defense
DT	Developmental Test
EA	Enterprise Architecture
EDT	Estimated Development Time
EVM	Earned Value Management
F&E	Facilities and Equipment
FAA	Federal Aviation Administration
FAR	Federal Acquisition Regulation
FFP	Firm Fixed Price
GAO	Government Accountability Office
GCSS-J	Global Combat Support System-Joint
IDIQ	Indefinite Delivery Indefinite Quantity
INCOSE	International Council on Systems Engineering
IPT	Integrated Product Team
ISPD	Implementation Strategy Planning Document
IT	Information Technology
JRC	Joint Resources Council
LeSS	Large-Scale Scrum
NAS	National Airspace System
NASA	National Aeronautics and Space Administration
O&M	Operations and Maintenance
OMB	Office of Management and Budget
OT	Operational Test
OT&E	Operational Test and Evaluation
PM	Project Manager
ROM	Rough Order of Magnitude
SAFe	Scaled Agile Framework
SEDC	Systems Engineering Development Conference
SEI	Software Engineering Institute

Acronym	Definition
SoS	Scrum of Scrums
SOW	Statement of Work
SP	Story Point
T&E	Test and Evaluation
T&M	Time-and-Materials
TDC	Total Development Cost
TDD	Test-Driven Development
TEMP	Test and Evaluation Master Plan
TSP	Total Story Points
U.S.	United States
US	User Story
VA	Department of Veterans Affairs

For Additional Information

For additional information or questions on the content provided by this document, please contact:

Avinash Pinto

avinash@mitre.org

(703) 983-4318